Technical University of Munich
TUM School of Computation, Information and Technology

Specification-Compliant Reachability Analysis for Motion Planning of Automated Vehicles

Edmond Irani Liu

Complete reprint of the dissertation approved by the TUM School of Computation, Information and Technology of the Technical University of Munich for the award of the

Doktor der Ingenieurwissenschaften (Dr.-Ing.).

The dissertation was submitted to the Technical University of Munich on 26.06.2023 and accepted by the TUM School of Computation, Information and Technology on 15.12.2023.

Abstract

Despite considerable efforts and progress in the development of automated vehicles, significant challenges persist that impede their mass deployment on public roads, among which is their motion planning modules. During every planning cycle, from potentially infinite driving trajectories, automated vehicles must decide on the ones that are dynamically feasible, collision-free, and compliant with specifications including mandatory traffic rules and supplementary handcrafted rules. The stringent real-time constraints of automated vehicles render exhaustive exploration and verification of individual trajectory candidates impossible; thus, the question arises of how to efficiently and reliably generate driving trajectories with the abovementioned properties. When extended to cooperative driving of a group of automated vehicles, this question becomes even more challenging since the number of trajectory combinations increases exponentially with respect to the number of vehicles within the group.

This dissertation proposes novel approaches centered around reachability analysis to address the problems of individual and cooperative specification-compliant motion planning. The core idea of this dissertation involves computing the collision-free reachable sets of automated vehicles, identifying their specification-compliant subsets, and extracting motion planning constraints to expedite the generation of trajectories adhering to specifications. Our methods (a) efficiently compute the drivable and specification-compliant reachable sets as the planning space for trajectory planning of individual and cooperative automated vehicles, and (b) exhaustively identify specification-compliant driving corridors within the computed reachable sets and determine the optimal ones among them. In addition, we integrate reachable sets with the so-called *invariably safe sets* for generating trajectories of cooperative vehicles such that their safety can be ensured for an infinite time horizon, even in safety-critical situations. As part of our contributions to the scientific community, we provide *CommonRoad-Reach*, an open-source toolbox for computing the reachable sets and extracting the driving corridors for automated vehicles with real-time capability. Furthermore, our *Scenario Factory* toolbox automatically generates distinct and safety-critical traffic scenarios from road networks across the globe, with which motion planners can be tested and compared. Our approaches' applicability, effectiveness, and efficiency are exhibited with experiments using various traffic scenarios from the publicly available CommonRoad benchmark suite.

Zusammenfassung

Trotz erheblicher Anstrengungen und Fortschritte bei der Entwicklung von automatisierten Fahrzeugen bestehen nach wie vor bedeutende Herausforderungen, die ihrer massenhaften Einsatz auf öffentlichen Straßen behindern, darunter ihre Bewegungsplanungsmodule. Während jedes Planungszyklus müssen automatisierte Fahrzeuge aus potenziell unendlichen Fahrtrajektorien solche auswählen, die dynamisch umsetzbar, kollisionsfrei und den Spezifikationen entsprechend sind, einschließlich der vorgeschriebenen Verkehrsregeln und zusätzlichen maßgeschneiderten Regeln. Die strengen Echtzeitbeschränkungen automatisierter Fahrzeuge machen eine umfassende Exploration und Überprüfung einzelner Trajektkandidaten unmöglich; somit stellt sich die Frage, wie Fahrtrajektorien mit den oben genannten Eigenschaften effizient und zuverlässig generiert werden können. Wenn dies auf das kooperative Fahren einer Gruppe von automatisierten Fahrzeugen erweitert wird, wird diese Fragestellung noch anspruchsvoller, da die Anzahl der Trajektkombinationen exponentiell mit der Anzahl der Fahrzeuge in der Gruppe zunimmt.

Diese Dissertation schlägt neuartige Ansätze vor, die sich um die Erreichbarkeitsanalyse von automatisierten Fahrzeugen drehen, um die Probleme der individuellen und kooperativen spezifikationskonformen Bewegungsplanung anzugehen. Die Kernidee dieser Dissertation besteht darin, die kollisionsfreien erreichbaren Mengen von automatisierten Fahrzeugen zu berechnen, ihre spezifikationskonformen Teilmengen zu identifizieren und Bewegungsplanungsbeschränkungen zu extrahieren, um die Generierung von spezifikationskonformen Trajektorien zu beschleunigen. Die vorgeschlagenen Methoden (a) berechnen effizient die befahrbaren und spezifikationskonformen erreichbaren Mengen als Planungsraum für die Trajektplaung von individuellen und kooperativen automatisierten Fahrzeugen und (b) identifizieren umfassend spezifikationskonforme Fahrkorridore innerhalb der berechneten erreichbaren Mengen und bestimmen die optimalen unter ihnen. Darüber hinaus integrieren wir erreichbare Mengen mit den sogenannten *unveränderlich sicheren Mengen*, um Trajektorien von kooperativen Fahrzeugen zu generieren, bei denen deren Sicherheit auch in sicherheitskritischen Situationen für einen unendlichen Zeithorizont gewährleistet werden kann. Als Beitrag zur wissenschaftlichen Gemeinschaft stellen wir *CommonRoad-Reach* vor, ein Open-Source-Toolbox zur Berechnung der erreichbaren Mengen und Extraktion der Fahrkorridore für automatisierte Fahrzeuge mit Echtzeitfähigkeit. Darüber hinaus generiert unsere *Scenario Factory*-Toolbox automatisch unterschiedliche und sicherheitskritische Verkehrsszenarien aus Straßennetzen weltweit, mit denen Bewegungsplaner getestet und verglichen werden können. Die Anwendbarkeit, Effektivität und Effizienz unserer Ansätze werden anhand von Experimenten mit verschiedenen Verkehrsszenarien aus der öffentlich verfügbaren CommonRoad Benchmark-Suite gezeigt.

Acknowledgment

This dissertation represents the culmination of my research conducted at the School of Computation, Information and Technology (formerly known as the Informatik) of the Technical University of Munich between the years 2019 and 2023. As I reflect upon the journey that has led me to this point, I would like to appreciate the support and contributions of many people.

First and foremost, I would like to express my utmost gratitude to my esteemed supervisor, Prof. Matthias Althoff, for providing me with the opportunity to commence my doctoral journey. His exceptional expertise, rigorous attitude to research, and constant encouragement throughout the years have been integral in shaping this dissertation. I consider myself fortunate to have had Matthias as my Ph.D. supervisor.

My heartfelt thanks extend to my colleagues from the Cyber-Physical Systems group, particularly the members of the CommonRoad team, as well as my friends within the academic community. Thank you all for the friendly and collaborative atmosphere within the group, as well as the memorable times we shared at the chair and the CPS workshops. I would like to say a special thank you to Xiao Wang, who provided me with assistance with various aspects of life in Germany. I am also thankful to Yuanfei Lin, with whom I engaged in many insightful discussions and who was my gym buddy during my stay at the GLC. I am truly grateful to have been surrounded by such a supportive community.

I would also like to acknowledge the students whose theses, seminars, or practical courses I had the chance of supervising, as well as the HiWis who provided valuable support to my work. Furthermore, I extend my appreciation to the secretaries of the chair Ute Lomp, Amy Bücherl, and Katja Hempel for their meticulous assistance with organizational matters. The generous financial support from the German Research Foundation and Huawei Technologies, as well as the fruitful collaborations with project partners, are gratefully acknowledged.

My deepest appreciation goes to my wife, Ke, for her unwavering support and trust throughout the past decade. Despite the difficulties imposed by the Covid pandemic, which kept us 9034 km apart for 1093 days, her love, understanding, and belief in me never faltered. She has been an immense source of inspiration and strength for me, and I am forever grateful for her presence in my life.

I would also like to extend my thanks to my family members (my two parents, three dogs, and four brothers), as their continuous encouragement and belief in my abilities have been crucial in my successful pursuit of higher educations.

In addition, I want to take a moment to acknowledge myself for embracing the challenges and committing to this arduous endeavor. The pursuit of a Ph.D. has demanded perseverance, resilience, and self-belief. Although far from perfect, I am proud of the efforts I have put into this research and cherish the personal growth along the way.

Lastly, I would like to thank the majestic Alps for providing me with moments of solace and excitement. The breathtaking beauty of the Alps (all seasons alike) and the unforgettable snowboarding experiences have served as a much-needed escape during the intense and difficult periods of research.

Clearly, I am not able to mention everyone who has contributed to the realization of this dissertation. Nevertheless, I would like to thank all of those who have supported me along the way in any shape or form. Thank you all for being a part of this significant milestone of my life.

Shenzhen, China, January 2024 Edmond Irani Liu

Those bluebird days. Stubaital, Austria, December 2022

Contents

Abstract iii

Zusammenfassung v

Acknowledgment vii

1 Introduction 1
- 1.1 Specification-Compliant Motion Planning 4
- 1.2 Cooperative Motion Planning 7
- 1.3 Contributions 9
- 1.4 Publications and Outline 11

2 Methods 13
- 2.1 General Setup 13
- 2.2 Temporal Logic 14
- 2.3 Reachability Analysis 16
- 2.4 Motion Planning using Reachability Analysis 19
- 2.5 Cooperative Motion Planning using Reachability Analysis 20

3 Conclusions 25
- 3.1 Summary 25
- 3.2 Future Work 26

Bibliography 29

A Reproduction of Publications 35
- A.1 Computing Specification-Compliant Reachable Sets for Motion Planning of Automated Vehicles [1] 36
- A.2 Specification-Compliant Driving Corridors for Motion Planning of Automated Vehicles [2] 45
- A.3 Specification-Compliant Motion Planning of Cooperative Vehicles Using Reachable Sets [3] 64

A.4 Provably-Safe Cooperative Driving via Invariably Safe Sets [4] 89
A.5 CommonRoad-Reach: A Toolbox for Reachability Analysis of Automated Vehicles [5] 98
A.6 Scenario Factory: Creating Safety-Critical Traffic Scenarios for Automated Vehicles [6] 107

B Supervised Theses **115**
B.1 Bachelor Theses 115
B.2 Master Theses 115

Chapter 1
Introduction

The advent of automated vehicles is envisioned to be the commencement of a new era of mobility, transforming, among others, the transportation and logistics industry in different ways. Owing to the immense potential of automated vehicles, they have received much attention in recent years from the academic and industrial sectors. According to a study by the U.S. national highway traffic safety administration in 2018 [7], the last event in the crash causal chain was assigned to human drivers in 94% of the crashes: The leading causes were recognition errors (41%), including driver's distractions and misidentifying surrounding objects, and decision errors (33%), such as over-speeding and ignoring traffic signs; about 7% of the crashes occurred due to drivers falling asleep. By replacing human drivers with sophisticated sensors, robust algorithms, and precise controllers, automated vehicles promise to offer numerous benefits, including (see Fig. 1.1):

- Enhanced safety: By drastically reducing, or ideally, eliminating traffic accidents, injuries, and fatalities through provably-correct systems, millions of lives could be saved, and immeasurable suffering could be prevented every year [8]. This is arguably the most significant benefit offered by automated vehicles.

- Increased efficiency: Automated vehicles' ability to quickly and optimally respond to dynamic environments significantly expands the overall road capacity and effectively mitigate traffic congestion. As a result, less travel time is required to reach destinations.

- Improved availability and accessibility: Without the need for human drivers, automated vehicles enable individuals that are dependent on others (e.g., children, the elderly, and those with physical restrictions or disabilities) to access mobility with less effort, possibly at any time.

Fig. 1.2 depicts EDGAR, one of the research vehicles developed at the Technical University of Munich equipped with the latest technologies.

These benefits become even more conspicuous when equipping automated vehicles with wireless technologies that facilitate Vehicle-to-Everything communication (see Fig. 1.3). As examples, information can be exchanged on the fly between automated vehicles, other traffic participants, and roadside infrastructure, e.g., to construct a comprehensive

Figure 1.1: Numerous benefits of automated vehicles over human drivers.

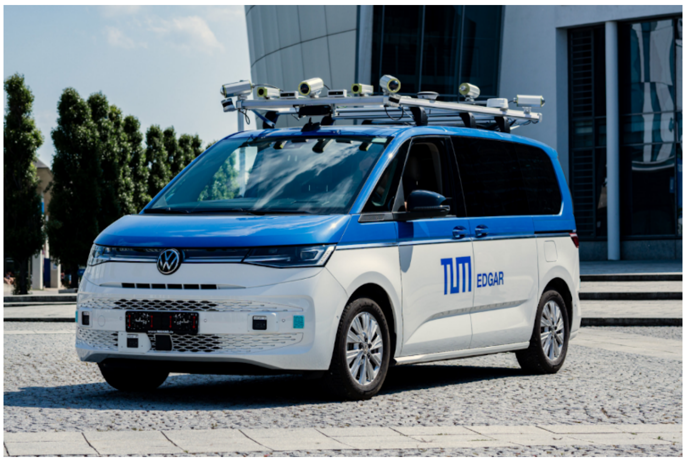
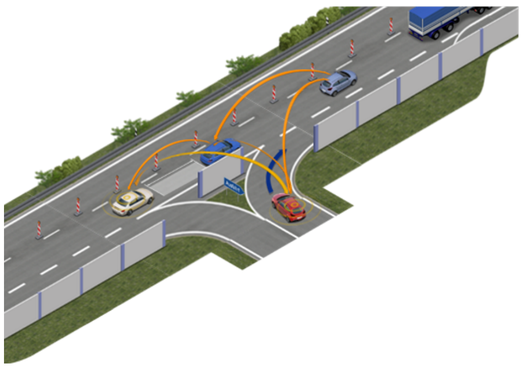


Figure 1.2: EDGAR: a research vehicle developed at the Technical University of Munich.

representation of the surrounding environment; also, the movements of multiple traffic participants can be coordinated more safely and efficiently, which is referred to as *maneuver-based* cooperation [9]. Consider, for instance, a group of connected automated vehicles. There are infinitely many traffic situations where cooperative motion planning of these vehicles increases their collective benefits; there are as many emergency situations where collisions with, e.g., reckless drivers performing sudden and illegal cut-ins, cannot be prevented without such cooperation.

Despite considerable efforts and progress in the field, substantial gaps and unresolved questions persist that impede the mass deployment of automated vehicles on public roads, in which automated and human-driven vehicles coexist. An accident[1] that took place in August 2023 greatly motivates the topic of this dissertation: A robotaxi operated by Cruise was making a right turn at an intersection while the traffic light was green,

[1]https://abc7news.com/cruise-driverless-car-sffd-fire-truck-accident/13666936/


(a) Cooperative perception

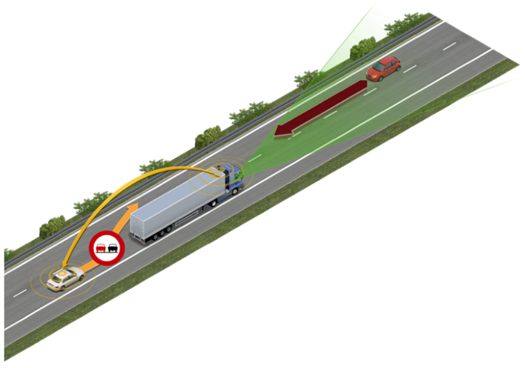

(b) Cooperative motion planning

Figure 1.3: Scenarios that benefit from cooperative driving. © Car2Car Communication Consortium

and it was collided by a fire truck on emergency mission. After careful investigation, the liability was assigned to the robotaxi. The reason was that, based on Californian traffic regulations, it is illegal not to yield the right of way to an emergency vehicle approaching with its lights and sirens. Unfortunately, the Cruise robotaxi did not handle this regulation well in its motion planning module, which ultimately led to the undesired collision.

Among others, the following research questions pertaining to the motion planning modules of automated vehicles remain open:

- Given arbitrary traffic scenarios with both static and dynamic obstacles, how can we efficiently and reliably determine the solution space of trajectories that are not only drivable, i.e., dynamically feasible and collision-free, but also compliant with specifications such as mandatory traffic rules?
- Within the abovementioned solution space, how can we exhaustively identify all driving maneuvers of the automated vehicles and determine the optimal ones among them?
- How can we generate drivable trajectories for automated vehicles that, too, comply with the enforced specifications?
- Can the approaches developed for individual automated vehicles be extended to groups of automated vehicles that drive cooperatively?
- Since safety is regarded as one of the most critical specifications for automated vehicles, how can we generate trajectories for a group of connected automated vehicles that maintain safety at all times, even in safety-critical situations?

This dissertation seeks to answer these open questions by developing methodologies and algorithms centered around the reachability analysis of automated vehicles, whose

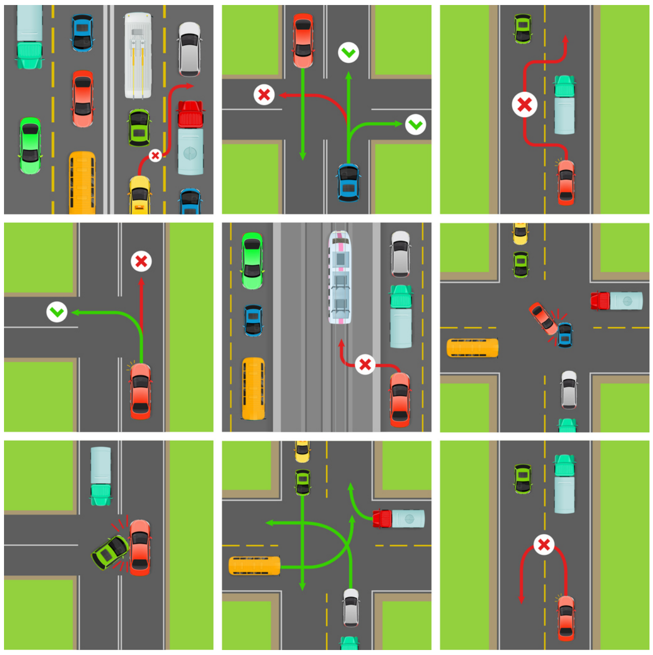


Figure 1.4: Complex scenarios with traffic rule violations (red trajectories). © iStock.com/robuart

preliminaries will be elaborated in Sec. 2.3. The following sections review existing literature on specification-compliant motion planning for individual automated vehicles and cooperative motion planning for groups of connected automated vehicles.

1.1 Specification-Compliant Motion Planning

In this dissertation, *specifications* refer to mandatory traffic rules and supplementary handcrafted rules to which automated vehicles should adhere. With the increasing presence of automated vehicles on public roads, it is imperative that they comply with the same specifications as human drivers in a precise and dependable manner. Undoubtedly, such compliance is essential for their safe and effective integration into road traffic. Furthermore, manufacturers of automated vehicle are responsible for certifying such compliance before mass deployment and by this avoid potential liability claims in undesired events such as car accidents. Fig. 1.4 depicts various traffic scenarios with traffic rule violations. Despite the significance of this matter, the majority of prior studies on the motion planning of automated vehicles, as reported in recent surveys [10]–[13], either entirely disregard specifications or only handles a limited subset. This circumstance can be attributed to the sheer difficulty of formalizing specifications in a machine-interpretable way and their integration into motion planners. In the following sections, we review the works proposed in the literature to tackle these challenges.

1.1.1 Specifications

A general guideline for participating in road traffic was established in the Vienna Convention on Road Traffic (VCoRT) back in the 1960s [14]. Many countries have adopted VCoRT as a basis for legislating national traffic rules, such as the German road traffic regulation. Existing traffic rules are typically expressed in the form of legal text, which are often imprecise, open to human interpretation, and not directly machine-interpretable. A set of concretized and formalized traffic rules is therefore essential to eliminate ambiguity in the interpretation and to be precisely evaluated on planned motions of automated vehicles. Relatively straightforward rules, such as maintaining lane-specific speed limits, can be easily incorporated as motion planning constraints (in this case, in the velocity domain of automated vehicles); however, more complex rules specifying temporal properties necessitate intricate formulations to capture the temporal constraints over motions of automated vehicles. Traffic rules can be formalized and expressed in several forms, including using ontology [15], [16], propositional logic [17], and temporal logic [18]–[22]. The former two typically require implicit embedding of temporal constraints in atomic propositions to reflect temporal relations between entities in a system. In contrast, one can explicitly and naturally describe such relations through well-defined temporal logic connectives (later detailed Sec. 2.2).

Commonly used temporal logics for linear-time specifications (as opposed to branching-time specifications) include linear temporal logic (LTL) [23], metric temporal logic (MTL) [24], signal temporal logic (STL) [25], and their numerous variants. LTL was initially proposed as a specification language for concurrent programs and perhaps is the most well-studied logic of these three. This prominence is partially due to its relative simplicity and its significance in the fields of automata theory and model checking; MTL extends LTL by introducing time intervals over temporal connectives so that specifications involving precise timing constraints can be effectively described; STL, on the other hand, is designed for reasoning about signals over time. The extent of satisfaction of specifications can be quantified, often referred to as the *robustness degree* [26].

Traffic rules from various sources have been formalized using different types of temporal logics. For instance, subsets of VCoRT have been formalized using LTL: One study [27] codified and concretized a set of rules on overtaking maneuvers, while other works [22], [28] formalized rules related to overtaking maneuvers, driving priorities, and coping with crossing pedestrians. To precisely capture timing constraints, articles [18], [21] utilize MTL to formalize German traffic rules covering interstate, intersection, and general driving situations. Similarly, article [19, Tab. 7] suggests that MTL is expressive enough to cover up to 95% of existing Chinese traffic rules. In work [29], the authors extend work [21] by reformulating traffic rules in STL and defining the robustness degree for relevant predicates. Apart from (inter)nationally-approved legal sources, researchers have also attempted to formalize rules that describe safe driving behaviors for automated vehicles. For example, the work in [30] encodes so-called *Responsibility-Sensitive Safety* (RSS) [31] rules in STL for monitoring purposes. In [32], the author presents a safety contract for automated vehicles in STL in which the lateral contracts differ from the rules described in RSS.

1.1.2 Specification-Compliant Motion Planning

Existing works on specification-compliant motion planning can be categorized based on *when* specifications are considered. We review each category in the following sections.

1.1.2.1 Considering Specifications After Motion Planning

Runtime verification, also known as *monitoring*, involves the inspection of whether executions of a system demonstrate expected behaviors. This technique is advantageous when the system under examination is either too complex to model or whose internal details are not easily accessible. As examples, work [27] presents a monitor for assessing compliance of vehicles with safe distance and overtaking rules, while work [30] describes a monitor for RSS rules.

Although the monitoring process can be executed efficiently, monitors typically only return a robustness degree of the specifications or a binary verdict, i.e., *true* or *false*, on whether the specifications have been satisfied by the system. In the context of motion planning for automated vehicles, monitors do not return alternative trajectories if a trajectory under inspections is deemed inferior or rejected. This often leads to replanning and verifying a large amount of trajectories before finding a specification-compliant solution, especially for more complex specifications.

An alternative to examining individual trajectories is to verify infinitely many trajectories at once: A method for model checking reachable sets of continuous and hybrid systems against STL specifications is described in [33]. However, as with monitoring, this method only returns a verdict, possibly together with counterexamples in case of violation, which has limited usage for motion planning for automated vehicles.

1.1.2.2 Considering Specifications During Motion Planning

The reviewed approaches in this category can be broadly classified into three groups: multilayered approaches, approaches based on mixed-integer linear programming (MILP), and approaches based on rapidly exploring random trees (RRTs) [34]. We describe their characteristics and shortcomings below.

Multilayered approaches [28], [35]–[43] typically handle the specification-compliant motion planning problem by employing an architecture consisting of a high-level discrete planning layer and a low-level trajectory planning layer. The discrete planning layer generates plans that satisfy considered specifications based on discrete abstractions of the system of interest, whereas the low-level trajectory planning layer outputs trajectories that adhere to the generated discrete plans. The discrete plans are generated based on, among others, automata theory [36], [38]–[41], [43], satisfiability modulo theory [35], [37], and monitors [28]. For example, article [40] generates timed paths that satisfy MTL specifications for indoor robot navigation using timed automata; article [37] introduces a satisfiability modulo convex programming framework that handles both convex constraints over continuous states and Boolean constraints over discrete states for cyber-physical systems; in article [28] the authors utilize monitors to obtain high-level driving maneuvers for automated vehicles that adhere to traffic rules expressed in LTL. In most cases, the discrete plans do not account for the dynamic constraints over the system of interest; therefore, the drivability of the discrete plans is typically not ensured. This fact may lead to frequent replanning in both the discrete and trajectory planning layers especially in complex and highly dynamic environments such as public roads.

MILP-based approaches commonly cast temporal logic specifications as mixed-integer linear constraints alongside system dynamic constraints. A solver is then employed to generate the optimal specification-compliant trajectory while optimizing certain user-defined cost functions. The unfavorable NP-hard nature of MILP problems [44, Ch. 11] and the exponential increase in complexity and solution time (e.g., see [45]–[49]) due to the auxiliary decision variables resulting from the enforced constraints are often limiting factors for applications with high real-time requirements. Moreover, MILP-based approaches generally require linear formulations of dynamic constraints, rendering integration of high-fidelity nonlinear vehicle models nontrivial. Both facts impede the adoption of MILP-based approaches for specification-compliant motion planning of automated vehicles.

The basic idea of RRT-based approaches is to plan specification-compliant trajectories in an incremental manner by spanning one or multiple trees exploring the state space of automated vehicles. Articles [38], [50]–[54] leverage the RRT* algorithm [55], which is an asymptotically optimal variant of the well-known RRT algorithm. The growth of the trees is steered or pruned, e.g., using automata [38], [50], [52], [54] or robustness degrees [51], [53] of specifications. If a trajectory adhering to the system's dynamic constraints and the considered specifications exists, it can be found given sufficient time and iterations. While RRT-based methods are capable of producing fast solutions to particular problems, their suitability for safety-critical applications is hindered by their inherent characteristic known as *probabilistic completeness* [34], [55]. Furthermore, the efficacy of RRT-based methods tends to deteriorate in situations where the solution space is narrowed and restricted [56]. For these reasons, RRT-based methods are not ideal for specification-compliant motion planning of automated vehicles.

1.2 Cooperative Motion Planning

As motivated at the beginning of the chapter, the benefits of automated vehicles fully unfold when they are able to communicate with other traffic participants: human drivers predominantly interact with each other through implicit communication and by anticipating the most likely behaviors of other traffic participants; conversely, when automated vehicles employ maneuver-based cooperation through explicit communications, they reduce uncertainties in future movements of participating vehicles and jointly offer more sophisticated and efficient solutions in an ongoing traffic scenario. Despite its advantages, cooperative motion planning is inherently a highly combinatorial problem whose complexity grows exponentially in relation to the number of participating vehicles (see

Sec. 2.5 and Fig. 2.5a). In addition, the cooperating vehicles must reach a consensus on their maneuvers in a timely manner while accounting for individual objectives and maintaining collision-free interactions with both non-communicating traffic participants and other members of the cooperative group. Therefore, a primary challenge of cooperative motion planning is the development of a computationally efficient cooperation scheme that upholds the optimality of the cooperative plans. Several recent surveys [57]–[61] provide comprehensive reviews of a plethora of works focusing on different aspects of cooperative driving of automated vehicles, including their architectures, maneuver planning schemes, and common use cases. Below, we briefly outline approaches relevant to cooperative motion planning of connected automated vehicles.

1.2.1 Multi-Objective Optimization

The cooperative motion planning of connected automated vehicles can be formulated as multi-objective optimization problems [57], [58], [61]. In optimization-based approaches, one or multiple optimization problems are formulated to minimize particular cost functions or, alternatively, to maximize certain utility functions. The cost and utility functions are designed to account for local and global objectives: Examples of local objectives are dynamic constraints, collision-avoidance constraints, and the preferences of individual vehicles; global objectives include the fairness of cooperation and the overall traffic efficiency of participating vehicles. These optimization problems are then solved using either centralized or distributed optimization techniques, generating reference trajectories to be followed by every cooperating vehicles. Despite offering viable solutions to the cooperative motion planning problems, optimization-based approaches generally suffer from the high computational complexity arising from an increased number of participating vehicles, either demanding extraordinary computational resources or imposing strict limitations on the size of the vehicle group.

1.2.2 Reservation-based Conflict Resolution

An alternative paradigm for solving cooperative motion planning problems is to adopt reservation-based methods [57], [58], [61]. The underlying concept of these methods is to treat free space on the road as a shared resource among cooperative vehicles and grant exclusive access to certain portions of the road for a limited duration to individual vehicles. By receiving these space-time allocations, cooperative vehicles can independently plan their motions and resolve potential collisions with other participating vehicles in the group. The minimal unit of reservation varies in the existing literature and can be represented by, e.g., tiles and cells [62], [63], conflicting points and regions [64], [65], and moving space-time corridors corresponding to specific driving maneuvers [66], [67].

The reservation can be performed in centralized or decentralized fashions. In centralized approaches, a central unit, such as a roadside infrastructure or a vehicle within the cooperative group, coordinates the maneuvers of all participating vehicles; whereas in decentralized approaches, participating vehicles initiate the cooperation and undertake actions on their own. We provide one example for both approaches: Article [62] serves as an illustration of earlier works that adopt centralized intersection management. Intersection regions are divided into tiles, which can be requested by vehicles approaching from different directions. Utilizing a first-come, first-served protocol, the intersection manager allocates tiles to vehicles while ensuring that no tile is simultaneously occupied by multiple vehicles. Articles [66], [67] exemplify fully decentralized approaches: An efficient and explicit space-time reservation protocol was devised for cooperative maneuver planning, with further real-world experiments reported in [68]. Cooperative vehicles may use this protocol to broadcast their desired space-time requests to nearby vehicles. Upon receiving approval from all surrounding vehicles whose motion would be directly influenced by the requests, the requesting vehicle obtains permission to navigate within the designated space-time envelope. While centralized approaches may readily generate globally optimal trajectories for all vehicles, they often confront the same high computational complexity associated with optimization-based approaches detailed in Sec. 1.2.1. In contrast, decentralized approaches impose less computational demands on each participant and exhibit greater flexibility, albeit at the expense of settling for suboptimal solutions.

A variety of policies can be enforced during the allocation of reservation units, ranging from the simpler first come, first served policy to the more intricate auction-based policies. The inefficiencies of the former policy, especially in high traffic density scenarios, can be mitigated by auction-based policies as demonstrated in [63], [69], [70]. Within auction-based frameworks, cooperative vehicles act as bidders competing for packages consisting of conflicting reservation units. An auction algorithm is executed to identify the package winners while optimizing the total revenue of the packages. Auction-based approaches offer notable advantages as they facilitate the systematic consideration of both individual and collective objectives and preferences concerning the packages.

1.3 Contributions

This dissertation proposes novel methods based on reachability analysis to address the (cooperative) specification-compliant motion planning problem for automated vehicles. As an attempt to answer the open questions described in page 4, the proposed methods [1]–[6] present the following contributions:

1. Efficient incorporation of specifications expressed in propositional logic and temporal logic into reachability analysis of automated vehicles, which yields specification-compliant planning spaces that can be used for various applications of automated vehicles, including motion planning and predicting the legal behaviors of other vehicles.

2. Efficient and exhaustive identification of drivable (dynamics-aware and collision-free) and specification-compliant driving corridors within the reachable sets of automated vehicles, among which the optimal choices are further determined. Applying constraints extracted from such driving corridors to motion planners expedites the generation of trajectories complying with enforced specifications.

3. Extension of the individual specification-compliant motion planning to a group of cooperative automated vehicles. This is achieved through negotiating their specification-compliant reachable sets. Consequently, each vehicle receives its own negotiated and specification-compliant solution space, within which trajectories can be individually planned.

4. Integration of the so-called invariably safe sets with cooperative motion planning and the generation of safe trajectories for cooperative vehicles such that their safety can be ensured for an infinite time horizon, even in safety-critical situations.

5. Provision of CommonRoad-Reach, an open-sourced toolbox written in Python and C++ that offers efficient and user-friendly implementations for computing the reachable sets and extracting the driving corridors of automated vehicles.

6. A pipeline for automatically generating a large amount of safety-critical traffic scenarios based on reachability analysis, which we refer to as Scenario Factory. The generated traffic scenarios are beneficial for various automated vehicle applications, including the testing, verification and comparison of motion planning algorithms. For example, our pipeline has been used to generate challenge scenarios in the international CommonRoad motion planning competition (years 2021 and 2022).

The underlying idea across [1]–[6] is to efficiently compute and represent the collision-free (and specification-compliant) solution space of automated vehicles for motion planning. Our reachability analysis exhibits numerous favorable properties, including:

1. Due to our reachability analysis' set-based and over-approximative nature, it efficiently explores the collision-free (and specification-compliant) solution space of automated vehicles that enclose all their drivable trajectories, even in scenarios with challenging narrow passageways.

2. Since we represent obstacles for collision checks by simple geometric shapes such as polygons and circles, our reachability analysis can be applied to complex traffic scenarios involving static and dynamic traffic participants of arbitrary shapes.

3. Contrary to conventional motion planners, our set-based reachability analysis requires less computation effort in more critical scenarios with shrinking solution space for motion planning.

4. The specification-compliant driving corridors identified within the computed reachable sets can be easily integrated into arbitrary motion planners accepting position and velocity constraints over the states of automated vehicles. In addition, such driving corridors do not restrict the vehicle models used for motion planning. For instance, they can be utilized by motion planners considering both coupled and decoupled longitudinal and lateral vehicle dynamics.

5. By coupling reachability analysis with model checking techniques, conflicting or non-satisfiable specifications can be detected prior to actually planning a low-level

trajectory. Moreover, it allows one to determine the last point in time at which a specification-compliant state still exists.

1.4 Publications and Outline

This dissertation is based on six peer-reviewed, (co-)first-author publications [1]–[6] that have been submitted to internationally acknowledged conferences, journals, or book publishers:

[1] **E. Irani Liu** and M. Althoff, "Computing specification-compliant reachable sets for motion planning of automated vehicles," in *Proc. of the IEEE Intell. Veh. Symp.*, 2021, pp. 1037–1044.

[2] **E. Irani Liu** and M. Althoff, "Specification-compliant driving corridors for motion planning of automated vehicles," *IEEE Trans. Intell. Veh.*, vol. 8, no. 9, pp. 4180–4197, 2023.

[3] **E. Irani Liu** and M. Althoff, "Specification-compliant motion planning of cooperative vehicles using reachable sets," in *Cooperatively Interacting Vehicles*, Springer, 2023.

[4] **E. Irani Liu**, C. Pek, and M. Althoff, "Provably-safe cooperative driving via invariably safe sets," in *Proc. of the IEEE Intell. Veh. Symp.*, 2020, pp. 516–523.

[5] **E. Irani Liu***, G. Würsching*, M. Klischat, and M. Althoff, "CommonRoad-Reach: A toolbox for reachability analysis of automated vehicles," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2022, pp. 2313–2320.

[6] M. Klischat*, **E. Irani Liu***, F. Höltke, and M. Althoff, "Scenario factory: Creating safety-critical traffic scenarios for automated vehicles," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2020, pp. 1–7.

* *These authors contributed equally to the article and share the first authorship.*

The reprint of each publication is included in this dissertation together with a summary of the content and the main contributions of E. I. L., the author of this dissertation, as required by the regulations of the degree-awarding institution. The contributions of the co-authors are not listed; nevertheless, their indispensable efforts in publishing the articles are gratefully acknowledged.

The remainder of this dissertation is structured as follows: Chapter. 2 presents the required preliminaries and general methodologies of our proposed methods. Chapter. 3 includes a summary of this dissertation and suggests possible directions for further improvements and future research. Appendix A contains the reprints of the included publications, and Appendix B lists the bachelor and master theses supervised by the author of this dissertation.

Chapter 2
Methods

This chapter introduces the necessary preliminaries and methodologies, including the general setup, temporal logics to formalize our specifications, set-based reachability analysis of automated vehicles, the concept of driving corridors, and (cooperative) motion planning of automated vehicles using reachability analysis.

2.1 General Setup

Throughout this dissertation, the vehicle for which trajectories should be planned is referred to as the *ego vehicle*. The ego vehicle receives as input the current environment model, including a road network, a local curvilinear coordinate system F^{L}, all relevant obstacles, and a set $\mathcal{F}$ of specifications. Fig. 2.1 illustrates an exemplary traffic scenario:

- The road network consists of lanelets [71], each of which has left and right boundaries modeled with polylines. A lanelet might reference traffic rule elements such as traffic signs and traffic lights. A route through the road network is planned via a readily available route planner from the initial state to the goal region, whose centerline is used as the reference path $\Gamma : \mathbb{R} \rightarrow \mathbb{R}^2$ of the ego vehicle.

- A local curvilinear coordinate system F^{L} is constructed from the reference path Γ using the method described in [71]. Adopting such a coordinate system facilitates the formulation of driving maneuvers from an ego-vehicle-centric perspective. Examples of such maneuvers include following a lane and stopping before an intersection. Within F^{L}, a tuple (s, d) describes the longitudinal coordinate s along Γ and the lateral coordinate d orthogonal to $\Gamma(s)$. It is worth noting that our computation of reachable sets (detailed in Sec. 2.3) is not restricted to F^{L} and can also be performed within the global Cartesian coordinate system where necessary, e.g., in scenarios without a structured road network such as parking lots.

- Relevant obstacles are those perceived within the field of view of the ego vehicle. It is assumed that the predicted motions of obstacles, e.g., their most likely trajectories, are given as input.

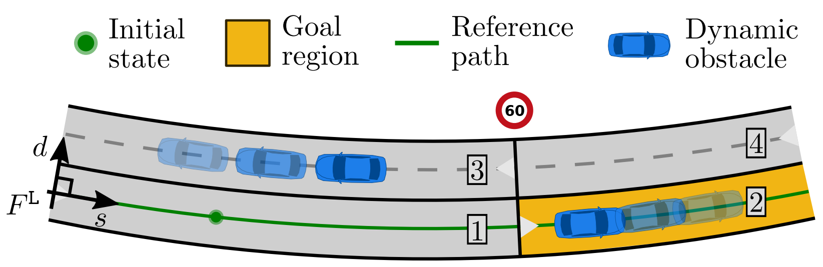


Figure 2.1: An exemplary traffic scenario with four lanelets (with IDs 1–4 shown in numbered boxes) and a curvilinear coordinate system constructed from the reference path. The triangles at the beginning of the lanelets indicate the driving directions. We also show the predicted motions of the vehicles for the future two steps with semi-transparency.

- The specifications $\mathcal{F}$ that we consider are expressed in temporal logic and will be introduced in Sec. 2.2.

The reachable sets and trajectories of the ego vehicle are computed at discrete steps $k \in \mathbb{N}_0$ corresponding to time $t_k = k\Delta_t$, with $\Delta_t \in \mathbb{R}_+$ being a predefined time increment, and up to the planning horizon $k_h \in \mathbb{N}$. The dynamics of the ego vehicle is

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k), \tag{2.1}$$

where $\boldsymbol{x}_k \in \mathcal{X}_k \subset \mathbb{R}^{n_{\boldsymbol{x}}}$ denotes the state of the ego vehicle in the state space $\mathcal{X}_k$, $\boldsymbol{u}_k \in \mathcal{U}_k \subset \mathbb{R}^{n_{\boldsymbol{u}}}$ denotes an input in the input space $\mathcal{U}_k$, all at step k. We use $\boldsymbol{U}$ to represent an input trajectory of the ego vehicle. Variable τ_k denotes the valuation (detailed in Sec. 2.2) of the ego vehicle with state $\boldsymbol{x}_k$ over a set $\mathcal{AP}$ of atomic propositions.

2.2 Temporal Logic

The specifications that we consider are expressed in *MTL with past over finite traces* (MTLp$_f$) [24], [72]. MTLp$_f$ shares the same syntax with MTL and is interpreted over traces of *finite* length (as opposed to *infinite* length in MTL). We adopt MTLp$_f$ specifications since (a) they are sufficiently expressive to formulate traffic rules with precise timing constraints, see, e.g., [18], [19], [21], and (b) traces in our system are of finite length. We now introduce MTLp$_f$ and its relevant subsets to our applications.

2.2.1 Metric Temporal Logic with Past over Finite Traces

An MTLp$_f$ formula $\varphi^{\mathtt{M}}$ over atomic propositions $\mathcal{AP}$ has the following syntax [24], [72]:

$$\varphi^{\mathtt{M}} ::= \sigma \,|\, \neg\varphi^{\mathtt{M}} \,|\, \varphi_1^{\mathtt{M}} \wedge \varphi_2^{\mathtt{M}}, \,|\, \mathbf{X}_I\varphi^{\mathtt{M}} \,|\, \varphi_1^{\mathtt{M}}\mathbf{U}_I\varphi_2^{\mathtt{M}} \,|\, \mathbf{Y}_I\varphi^{\mathtt{M}} \,|\, \varphi_1^{\mathtt{M}}\mathbf{S}_I\varphi_2^{\mathtt{M}},$$

where $\sigma \in \mathcal{AP}$ is an atomic proposition reflecting a logical relation between the ego vehicle and entities in the environment model, $\neg$ (Not) and $\wedge$ (And) are Boolean connectives, $\mathbf{X}$ (ne**X**t) and $\mathbf{U}$ (**U**ntil) are future-time connectives, $\mathbf{Y}$ (**Y**esterday) and $\mathbf{S}$

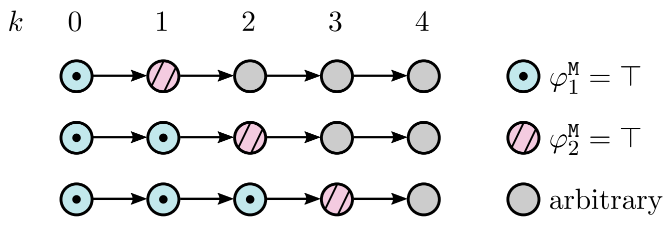
(**S**ince) are past-time connectives, and $I = [a, b]$, $a, b \in \mathbb{N}_0$, is a bounded interval. In addition, we also adopt the following commonly used abbreviations [24]:

- Contradiction: $\bot \equiv \varphi^{\mathtt{M}} \wedge \neg\varphi^{\mathtt{M}}$,
- Tautology: $\top \equiv \neg\bot$,
- Or: $\varphi_1^{\mathtt{M}} \vee \varphi_2^{\mathtt{M}} \equiv \neg(\neg\varphi_1^{\mathtt{M}} \wedge \neg\varphi_2^{\mathtt{M}})$,
- Implication: $\varphi_1^{\mathtt{M}} \Rightarrow \varphi_2^{\mathtt{M}} \equiv \neg\varphi_1^{\mathtt{M}} \vee \varphi_2^{\mathtt{M}}$,
- **F**uture: $\mathbf{F}_I\varphi^{\mathtt{M}} \equiv \top\mathbf{U}_I\varphi^{\mathtt{M}}$,
- **G**lobally: $\mathbf{G}_I\varphi^{\mathtt{M}} \equiv \neg\mathbf{F}_I\neg\varphi^{\mathtt{M}}$,
- **O**nce: $\mathbf{O}_I\varphi^{\mathtt{M}} \equiv \top\mathbf{S}_I\varphi^{\mathtt{M}}$,
- **H**istorically: $\mathbf{H}_I\varphi^{\mathtt{M}} \equiv \neg\mathbf{O}_I\neg\varphi^{\mathtt{M}}$.

We interpret MTLp$_f$ over the *point-wise* semantics [73] with timed traces, which resemble sequences of events with timestamps. Given is a trace $\tau := (\tau^0, \dots, \tau^k, \dots)$ with length $|\tau|$, where $\tau^k : \mathcal{AP} \to \{\mathtt{true}, \mathtt{false}\}$ represents a valuation over $\mathcal{AP}$, i.e., an assignment of `true` or `false` to every atomic proposition $\sigma \in \mathcal{AP}$, at step k. The notation $(\tau, k) \models \varphi^{\mathtt{M}}$ indicates that $\varphi^{\mathtt{M}}$ holds in the k-th valuation of τ, i.e., τ^k. Due to the reason that τ^k are synchronized with steps k, we simplify the semantics of MTLp$_f$ presented in [24],[72, Sec. 2]:

- $(\tau, k) \models \sigma$ if and only if (iff) $\tau^k(\sigma) = \mathtt{true}$,
- $(\tau, k) \models \neg\varphi^{\mathtt{M}}$ iff $(\tau, k) \not\models \varphi^{\mathtt{M}}$,
- $(\tau, k) \models \varphi_1^{\mathtt{M}} \wedge \varphi_2^{\mathtt{M}}$ iff $(\tau, k) \models \varphi_1^{\mathtt{M}}$ and $(\tau, k) \models \varphi_2^{\mathtt{M}}$,
- $(\tau, k) \models \mathbf{X}_I\varphi^{\mathtt{M}}$ iff $k < |\tau| - 1$, $1 \in I$, and $(\tau, k+1) \models \varphi^{\mathtt{M}}$,
- $(\tau, k) \models \mathbf{Y}_I\varphi^{\mathtt{M}}$ iff $k > 0$, $1 \in I$, and $(\tau, k-1) \models \varphi^{\mathtt{M}}$,
- $(\tau, k) \models \varphi_1^{\mathtt{M}}\mathbf{U}_I\varphi_2^{\mathtt{M}}$ iff $\exists l$, $k \leq l \leq |\tau| - 1$: $(\tau, l) \models \varphi_2^{\mathtt{M}}$, $l - k \in I$, and $\forall m$, $k \leq m < l$: $(\tau, m) \models \varphi_1^{\mathtt{M}}$,
- $(\tau, k) \models \varphi_1^{\mathtt{M}}\mathbf{S}\varphi_2^{\mathtt{M}}$ iff $\exists l$, $0 \leq l \leq k$: $(\tau, l) \models \varphi_2^{\mathtt{M}}$, $k - l \in I$, and $\forall m$, $l < m \leq k$: $(\tau, m) \models \varphi_1^{\mathtt{M}}$.

As examples, formulas $\mathbf{X}_{[2,4]}\varphi^{\mathtt{M}}$ and $\varphi_1^{\mathtt{M}}\mathbf{U}_{[1,3]}\varphi_2^{\mathtt{M}}$ respectively mean that "next valuation occurs within 2 and 4 steps from now, in which $\varphi^{\mathtt{M}}$ holds" and "within 1 and 3 steps, a valuation occurs in which $\varphi_2^{\mathtt{M}}$ holds, and $\varphi_1^{\mathtt{M}}$ holds for all valuations before that". Fig. 2.2 depicts traces that satisfy the latter formula. The past-time connectives **Y**, **S**, **O**, and **H** respectively mirror their future-time counterparts **X**, **U**, **F**, and **G**, backward in time.



k
0
1
2
3
4
$\varphi_1^{\mathtt{M}} = \top$
$\varphi_2^{\mathtt{M}} = \top$
arbitrary

Figure 2.2: Illustration of traces satisfying $\varphi_1^{\mathtt{M}} \mathbf{U}_{[1,3]} \varphi_2^{\mathtt{M}}$. A circle represents a valuation in which the atomic proposition corresponding to the color is assigned `true`. A sequence of circles represents a trace.

2.2.2 Linear Temporal Logic (with Past over Finite Traces)

Since valuations τ^k in our system are synchronized with steps k, we do not require the full expressiveness of MTLp$_f$ for checking the compliance of our system against the considered specifications. To reduce the computational complexity and to leverage existing model checkers, we interpret MTLp$_f$ formulas as *LTL with past over finite traces* (LTLp$_f$) [74] and further convert them into LTL over *infinite* traces. MTLp$_f$ is reduced to LTLp$_f$ by dropping intervals I over the temporal connectives [74] in the syntax; further dropping past-time connectives yields standard LTL [23]. We denote the set of specifications converted into LTL with which the ego vehicle should comply by $\mathcal{F}$.

2.2.3 Propositional Logic

The subset of MTLp$_f$ formulas without temporal connectives is essentially propositional logic [75]. Since propositional logic specifications do not reason about atomic propositions in terms of time, they are directly handled during the forward propagation of reachable set computation (see [1] for details).

2.3 Reachability Analysis

We apply set-based reachability analysis and compute the collision-free reachable sets of automated vehicles to explore their continuous state spaces over time. The computed reachable sets, or their specification-compliant subsets in case certain specifications $\mathcal{F}$ are to be considered, are advantageous for a series of applications, including restricting the motion planning space of automated vehicles. We introduce a few definitions and the computation of reachable sets.

Definition 1 (Occupancy)**.** *The operator* $\mathrm{occ}(\cdot)$ *returns the positions occupied within* F^{L}*. For example,* $\mathrm{occ}(\boldsymbol{x}_k)$ *returns the occupancy of the ego vehicle with state* $\boldsymbol{x}_k$*.*

Definition 2 (Set of forbidden states)**.** *Given the set* $\mathcal{O}_k \subset \mathbb{R}^2$ *of positions occupied by all obstacles at step k and the space outside the road, the set of forbidden states of the ego vehicle at step k is defined as*

$$\mathcal{X}_k^{\mathtt{F}} := \left\{ \boldsymbol{x}_k \in \mathcal{X}_k \,\middle|\, \mathrm{occ}(\boldsymbol{x}_k) \cap \mathcal{O}_k \neq \varnothing \right\}. \tag{2.2}$$

Definition 3 (One-Step Reachable Set)**.** *Let us introduce $\mathcal{R}^{\mathsf{e}}_0 = \mathcal{X}_0$ as the exact reachable set of the ego vehicle at the initial step, with $\mathcal{X}_0$ being the set of collision-free initial states including measurement uncertainties. The exact reachable set $\mathcal{R}^{\mathsf{e}}_{k+1}$ is the set of states reachable from $\mathcal{R}^{\mathsf{e}}_k$ within one step without intersecting the set of forbidden states $\mathcal{X}^{\mathsf{F}}_{k+1}$, denoted by* $\mathrm{reach}(\mathcal{R}^{\mathsf{e}}_k)$*:*

$$\mathcal{R}^{\mathsf{e}}_{k+1} := \underbrace{\Big\{\boldsymbol{x}_{k+1} \in \mathcal{X}_{k+1} \Big| \exists \boldsymbol{x}_k \in \mathcal{R}^{\mathsf{e}}_k, \exists \boldsymbol{u}_k \in \mathcal{U}_k : \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k), \boldsymbol{x}_{k+1} \notin \mathcal{X}^{\mathsf{F}}_{k+1}\Big\}}_{\mathrm{reach}(\mathcal{R}^{\mathsf{e}}_k)}. \tag{2.3}$$

Definition 4 (Projection)**.** *The operator* $\mathrm{proj}_{\Diamond}(\boldsymbol{x})$ *maps the state $\boldsymbol{x} \in \mathcal{X}$ to its elements $\Diamond$. For example,* $\mathrm{proj}_{(s,\dot{s})}(\boldsymbol{x}) = (s, \dot{s})^{\mathsf{T}}$ *for $\boldsymbol{x} = (s, \dot{s}, \ddot{s})^{\mathsf{T}}$. A set can be projected using the same operator:* $\mathrm{proj}_{\Diamond}(\mathcal{X}) = \big\{\mathrm{proj}_{\Diamond}(\boldsymbol{x}) \,\big|\, \boldsymbol{x} \in \mathcal{X}\big\}$.

Definition 5 (Drivable Area)**.** *The drivable area $\mathcal{D}^{\mathsf{e}}_k$ at step k is defined as the projection of the reachable set $\mathcal{R}^{\mathsf{e}}_k$ onto the position domain, i.e., $\mathcal{D}^{\mathsf{e}}_k :=$* $\mathrm{proj}_{(s,d)}(\mathcal{R}^{\mathsf{e}}_k)$.

In practice, the computation of the exact reachable set $\mathcal{R}^{\mathsf{e}}_k$ is generally difficult or even impossible for certain classes of systems [76]. In addition, computing the reachable sets is computationally challenging given the arbitrary shapes of the obstacles on the road, the high-dimensional continuous state space of a vehicle, its non-linear vehicle model, and its hard real-time constraints over motion planning. For these reasons, we instead compute tight over-approximations $\mathcal{R}_k \supseteq \mathcal{R}^{\mathsf{e}}_k$ of the reachable sets and drivable areas $\mathcal{D}_k \supseteq \mathcal{D}^{\mathsf{e}}_k$.

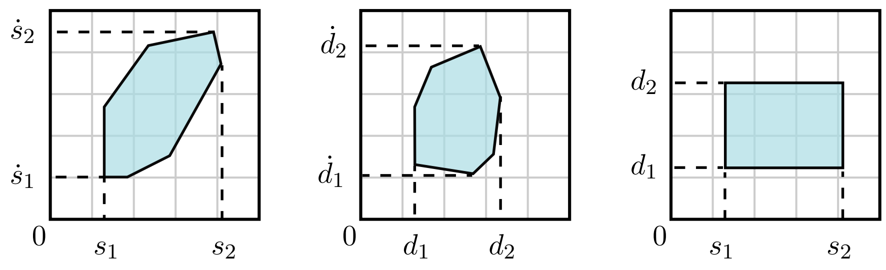
The dynamics of the ego vehicle (2.1) is abstracted by two double integrators within the coordinate system F^{L} for computational efficiency. For simplicity, the geometric center of the ego vehicle is chosen as the reference point. Our model has states $\boldsymbol{x}_k = (s_k, \dot{s}_k, d_k, \dot{d}_k)^{\mathsf{T}}$ and inputs $\boldsymbol{u}_k = (\ddot{s}_k, \ddot{d}_k)^{\mathsf{T}}$:

$$\boldsymbol{x}_{k+1} = \underbrace{\begin{pmatrix} 1 & \Delta_t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta_t \\ 0 & 0 & 0 & 1 \end{pmatrix} \boldsymbol{x}_k + \begin{pmatrix} \frac{1}{2}\Delta_t^2 & 0 \\ \Delta_t & 0 \\ 0 & \frac{1}{2}\Delta_t^2 \\ 0 & \Delta_t \end{pmatrix} \boldsymbol{u}_k}_{f(\boldsymbol{x}_k, \boldsymbol{u}_k)}. \tag{2.4}$$

This abstraction is a trade-off between the accuracy of capturing the actual behaviors of the ego vehicle and the computational effort required for its reachability analysis. Moreover, this abstraction ensures that the reachable sets of high-fidelity vehicle models are always contained within those of the adopted model. Let $\square$ denote a variable with its minimum and maximum values respectively represented by $\underline{\square}$ and $\overline{\square}$. The velocities and accelerations of the ego vehicle at (s_k, d_k) are bounded by

$$\underline{\dot{s}}(\Gamma, s_k) \leq \dot{s}_k \leq \overline{\dot{s}}(\Gamma, s_k),\ \underline{\dot{d}}(\Gamma, s_k) \leq \dot{d}_k \leq \overline{\dot{d}}(\Gamma, s_k), \tag{2.5a}$$

$$\underline{\ddot{s}}(\Gamma, s_k) \leq \ddot{s}_k \leq \overline{\ddot{s}}(\Gamma, s_k),\ \underline{\ddot{d}}(\Gamma, s_k) \leq \ddot{d}_k \leq \overline{\ddot{d}}(\Gamma, s_k). \tag{2.5b}$$



(a) $\mathcal{P}_k^{(i),s}$ **(b)** $\mathcal{P}_k^{(i),d}$ **(c)** $\mathcal{D}_k^{(i)}$

Figure 2.3: Polytopes and drivable area of a base set $\mathcal{R}_k^{(i)}$.

We choose these bounds conservatively to consider the kinematic limitations within a curvilinear coordinate system, see, e.g., article [77].

Following article [78], the occupancy of the ego vehicle $\mathrm{occ}(\boldsymbol{x}_k)$ in (2.2) is under-approximated by its inscribed circle, resulting in an under-approximative set of forbidden states $\check{\mathcal{X}}_k^{\mathsf{F}}$. Thus, the reachable sets $\mathcal{R}_k \supseteq \mathcal{R}_k^{\mathsf{e}}$ are over-approximative and enclose all drivable trajectories of the ego vehicle. To reduce the required computational effort, the reachable sets $\mathcal{R}_k$ are represented by the union of so-called *base sets* $\mathcal{R}_k^{(i)}$, $i \in \mathbb{N}$, i.e., $\mathcal{R}_k := \cup_i \mathcal{R}_k^{(i)}$. Every base set $\mathcal{R}_k^{(i)} := \mathcal{P}_k^{(i),s} \times \mathcal{P}_k^{(i),d}$ is a Cartesian product of two convex polytopes that respectively enclose the reachable positions and velocities of the ego vehicle in the $(s, \dot{s})$ and $(d, \dot{d})$ planes (see Fig. 2.3a–b). Compared to other set representations listed in [79, Tab. 1], this representation is advantageous since polytopes are closed under linear mappings and intersections. To simplify the notation, the collection[1] of $\mathcal{R}_k^{(i)}$ is also denoted by $\mathcal{R}_k$, i.e., $\mathcal{R}_k := \{\dots, \mathcal{R}_k^{(i)}, \dots\}$. The states $\boldsymbol{x}_k$ within a base set $\mathcal{R}_k^{(i)}$ have a unified valuation over atomic propositions $\mathcal{AP}$, denoted by $\tau_k^{(i)}$. Similar to Def. 5, the projections of sets $\mathcal{R}_k$ and $\mathcal{R}_k^{(i)}$ onto the position domain are denoted by $\mathcal{D}_k$ and $\mathcal{D}_k^{(i)}$, respectively, and $\mathcal{D}_k := \cup_i \mathcal{D}_k^{(i)}$ (see Fig. 2.3c).

For later use, the reachability between the collision-free base sets of consecutive steps is determined following the procedure described in [1], which is stored in a directed and acyclic *reachability graph* G^{R}. An edge $(\mathcal{R}_k^{(i)}, \mathcal{R}_{k+1}^{(j)})$ in graph G^{R} indicates that set $\mathcal{R}_{k+1}^{(j)}$ is reachable from set $\mathcal{R}_k^{(i)}$ after one step.

We use the reachable sets $\mathcal{R}_k$ to explore the collision-free solution space of the ego vehicle for motion planning. However, the direct usage of $\mathcal{R}_k$ is unsuitable for obtaining constraints for generating specification-compliant trajectories due to the reasons that $\mathcal{R}_k$ (a) may be disconnected in the position domain and (b) may contain states $\boldsymbol{x}_k$ having different valuations over atomic propositions. To overcome this problem, we construct collision-free connected components within $\mathcal{R}_k$ and further identify specification-compliant driving corridors, from which motion planning constraints over the states $\boldsymbol{x}_k$ can be extracted. We proceed with presenting relevant definitions.

[1]We refer to a set of sets as a collection.

Definition 6 (Connected Component). *A connected component $\mathcal{C}_k \subseteq \mathcal{R}_k$ is a set with valuation τ_k over atomic propositions $\mathcal{AP}$ such that*

(a) $\mathcal{C}_k$ forms a connected set [80] and is collision-free in the position domain,

(b) the states $\boldsymbol{x}_k$ within $\mathcal{C}_k$ have the same valuation τ_k.

Definition 7 (Driving Corridor). *A driving corridor is a sequence of connected components $(\mathcal{C}_0, \dots, \mathcal{C}_k, \dots, \mathcal{C}_{k_h})$ over steps k.*

To expedite the generation of specification-compliant trajectories, we also define driving corridors compliant with the specifications $\mathcal{F}$.

Definition 8 (Specification-Compliant Driving Corridor). *A specification-compliant driving corridor is one that complies with specifications $\mathcal{F}$: $\forall \varphi \in \mathcal{F} : (\tau_0, \dots, \tau_{k_h}) \models \varphi$.*

2.4 Motion Planning using Reachability Analysis

The specification-compliant motion planning problem of automated vehicles is formally defined as follows:

Problem 1 (Specification-Compliant Motion Planning). *Find an optimal control input $\boldsymbol{U}^*$ that solves the following non-convex optimization problem:*

$$\min_{\boldsymbol{U}} \sum_{k=0}^{k_h} J(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{2.6a}$$

$$\textit{subject to} \quad \boldsymbol{x}_0 = \tilde{\boldsymbol{x}}_0, \tag{2.6b}$$

$$\boldsymbol{x}_{k_h} \in \mathcal{X}^{\mathtt{G}}, \tag{2.6c}$$

$$\forall k \in \{0, \dots, k_h - 1\} : \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k), \tag{2.6d}$$

$$\forall k \in \{0, \dots, k_h\} : \boldsymbol{x}_k \notin \mathcal{X}_k^{\mathtt{F}}, \tag{2.6e}$$

$$\forall \varphi \in \mathcal{F} : (\tau_0, \dots, \tau_{k_h}) \models \varphi, \tag{2.6f}$$

where $J : \mathbb{R}^{n_{\boldsymbol{x}}} \times \mathbb{R}^{n_{\boldsymbol{u}}} \to \mathbb{R}$ is a cost function, $\tilde{\boldsymbol{x}}_0 \in \mathcal{X}_0$ is the measured initial state of the ego vehicle, the set $\mathcal{X}^{\mathtt{G}}$ denotes the goal states of the ego vehicle, and the set $\mathcal{F}$ represents the enforced specifications. Constraints (2.6e) *and* (2.6f) *respectively encode collision avoidance and specification compliance.*

A feasible solution to Prob. 1 can be obtained using the methods reviewed in Sec. 1.1.2, each with its potential drawbacks. To expedite the generation of solution trajectories, it is often beneficial, or sometimes even necessary [81], [82], to guide the ego vehicle in the form of additional constraints over its states $\boldsymbol{x}_k$. Therefore, we leverage our specification-compliant driving corridors (see Def. 8), from which we extract additional motion planning constraints. The optimal driving corridor adhering to specifications $\mathcal{F}$ is obtained by solving the following problem:

Problem 2 (Optimal Specification-Compliant Driving Corridor Identification)**.** *The optimal specification-compliant driving corridor of the ego vehicle is the solution to the following optimization problem:*

$$\max \sum_{k=0}^{k_h} u_k \tag{2.7a}$$

$$\textit{subject to} \quad \mathcal{C}_0 \ni \boldsymbol{x}_0, \tag{2.7b}$$

$$\mathcal{C}_{k_h} \cap \mathcal{X}^{\mathtt{G}} \neq \varnothing, \tag{2.7c}$$

$$\forall k \in \{0, \dots, k_h - 1\} : \mathcal{C}_{k+1} \cap \operatorname{reach}(\mathcal{C}_k) \neq \varnothing, \tag{2.7d}$$

$$\forall \varphi \in \mathcal{F} : (\tau_0, \dots, \tau_{k_h}) \models \varphi, \tag{2.7e}$$

where u_k is the utility associated with $\mathcal{C}_k$ (detailed in [2]). Constraints (2.7d) *and* (2.7e) *respectively encode the reachability of successive connected components of the driving corridor and its compliance with the enforced specifications. Collision-freeness of the driving corridor follows directly from Def. 6.*

Given a driving corridor within the reachable sets of the ego vehicle, additional constraints over its states $\boldsymbol{x}_k$ can be introduced to Prob. 1 such that the trajectory of the ego vehicle is contained within the driving corridor: $\forall k \in \{0, \dots, k_h\} : \boldsymbol{x}_k \in \mathcal{C}_k$. Let us describe the general procedure of our approach for obtaining a specification-compliant trajectory using reachable sets with the help of Fig. 2.4:

1. Explore the collision-free solution space for motion planning by computing the (over-approximative) reachable sets of the ego vehicle for consecutive steps up to the planning horizon (Fig. 2.4a).

2. Identify (all) collision-free and specification-compliant driving corridors within the computed reachable sets. In Fig. 2.4b–c, we illustrate two exemplary driving corridors within the reachable sets of the ego vehicle, with one being compliant and the other non-compliant with the enforced specifications.

3. Determine the (optimal) driving corridor in which low-level trajectories should be planned. Note that the number of possible driving corridors grows exponentially in relation to the planning horizon and there are potentially many more driving corridor candidates besides those depicted in Fig. 2.4b–c.

4. Plan a trajectory using arbitrary motion planners accepting constraints extracted from the connected components of the (optimal) driving corridor (Fig. 2.4d).

2.5 Cooperative Motion Planning using Reachability Analysis

Now let us consider a group of cooperative vehicles V_n, $n \in \mathcal{N} = \{1, 2, \dots, N\}$, whose associated variables $\square$ are denoted by $\square_n$. The primary goal of cooperative motion planning is to orchestrate the motions of the cooperative vehicles such that each vehicle follows a collision-free trajectory with respect to all other traffic participants. On top of this, we require that each vehicle adheres to its set of specifications $\mathcal{F}_n$. The cooperative motion planning problem can be formulated by extending Prob. 1:

Problem 3 (Cooperative Specification-Compliant Motion Planning)**.** *Find an optimal control input $\boldsymbol{U}_n^*$ for each cooperative vehicle V_n and solve the following non-convex optimization problem:*

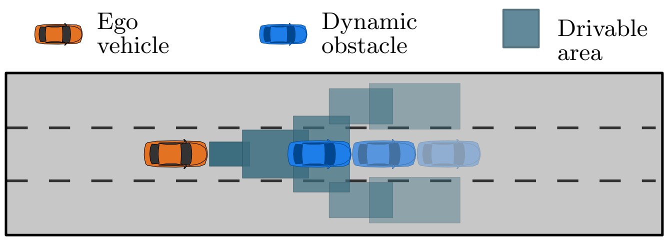
$$\min_{\hat{\boldsymbol{U}}} \sum_{n=1}^{N} \sum_{k=0}^{k_h} J_n(\boldsymbol{x}_{k,n}, \boldsymbol{u}_{k,n}) \tag{2.8a}$$

$$\begin{aligned}
\textit{subject to} \quad \forall n, m \in \mathcal{N} : \boldsymbol{x}_{0,n} &= \tilde{\boldsymbol{x}}_{0,n}, && (2.8b)\\
\boldsymbol{x}_{k_h,n} &\in \mathcal{X}_n^{\mathtt{G}}, && (2.8c)\\
\forall k \in \{0, \dots, k_h - 1\} : \boldsymbol{x}_{k+1,n} &= f_n(\boldsymbol{x}_{k,n}, \boldsymbol{u}_{k,n}), && (2.8d)\\
\forall k \in \{0, \dots, k_h\} : \boldsymbol{x}_{k,n} &\notin \mathcal{X}_{k,n}^{\mathtt{F}}, && (2.8e)\\
\mathrm{occ}(\boldsymbol{x}_{k,n}) \cap \mathrm{occ}(\boldsymbol{x}_{k,m}) &= \varnothing, n \neq m, && (2.8f)\\
\forall \varphi_n \in \mathcal{F}_n : (\tau_{0,n}, \dots, \tau_{k_h,n}) &\models \varphi_n, && (2.8g)
\end{aligned}$$

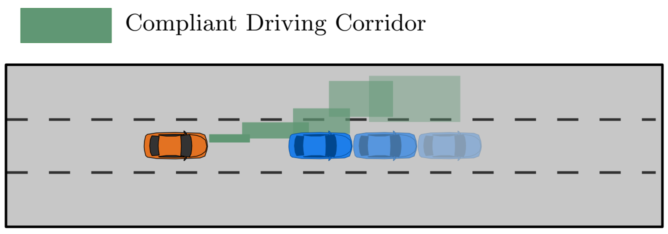
where $\hat{\boldsymbol{U}} = (\boldsymbol{U}_1, \boldsymbol{U}_2, \dots, \boldsymbol{U}_N)^\mathsf{T}$ is a vector of control inputs for each vehicle V_n over the planning horizon. Collision avoidance with non-cooperating traffic participants and with other cooperating vehicles within the group is respectively handled by (2.8e) *and* (2.8f)*.*

The complexity of trajectory-based approaches for solving Prob. 3 is exponential with relation to the number N of cooperative vehicles: Assuming that each vehicle generates M trajectory candidates, there are M^N possible combinations of trajectories to be evaluated (see Fig. 2.5a). We resort to a computationally efficient approach to overcome this issue: The permissible solution space of each vehicle is determined using reachability analysis, where potential conflicts in the position domain, i.e., collisions within the group, are detected and resolved. Consequently, each vehicle unambiguously receives its own collision-free planning space for trajectory planning. It is worth noting that cooperative vehicles can adopt different motion planning techniques to generate their trajectories within the negotiated planning space. The general procedure of our approach for obtaining specification-compliant trajectories for cooperative vehicles using reachable sets is as follows:

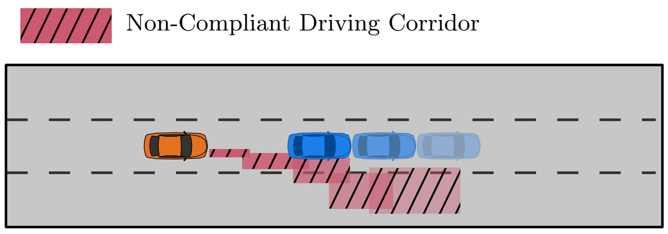
1. Each cooperative vehicle V_n computes its specification-compliant reachable set for one step, followed by negotiating conflicting reachable sets within the cooperative group. This process is repeated up to the planning horizon (see Fig. 2.5c).
2. Each vehicle V_n identifies (all) collision-free and specification-compliant driving corridors within its negotiated reachable sets.
3. Each vehicle V_n determines its (optimal) driving corridor in which trajectories should be planned.
4. Each vehicle V_n plans a trajectory using arbitrary motion planners accepting constraints extracted from its (optimal) driving corridor (see Fig. 2.5d).



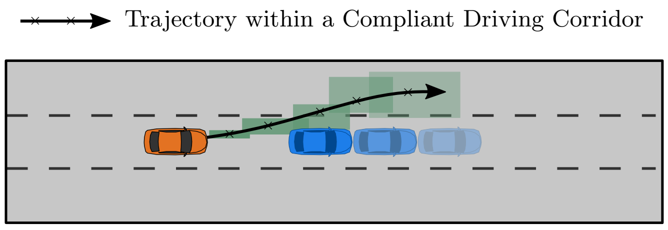
(a) Collision-free drivable areas of the ego vehicle over consecutive steps.



(b) A compliant driving corridor that overtakes the leading vehicle from its left side.

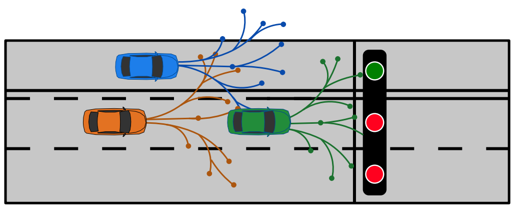


(c) A non-compliant driving corridor that overtakes the leading vehicle from its right side.

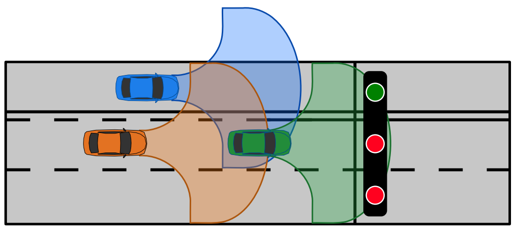


(d) A trajectory planned within a driving corridor using arbitrary motion planners.

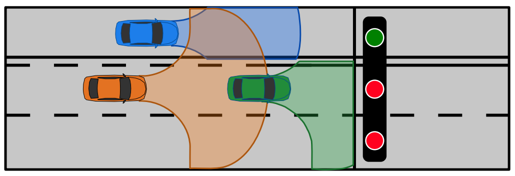
Figure 2.4: General procedure for obtaining a specification-compliant trajectory using reachable sets. In this scenario, the ego vehicle is trying to overtake its leading vehicle. We assume that the enforced specifications forbid overtaking a vehicle from its right side; thus, the ego vehicle plans a trajectory overtaking from the left side of its leading vehicle. Results of future steps are shown with increased transparency.



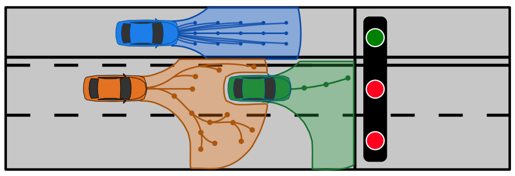
(a) Highly combinatorial nature of cooperative motion planning.



(b) Drivable areas of vehicles disregarding traffic rules.



(c) Drivable areas of vehicles considering traffic rules.



(d) Planned trajectories of vehicles within the drivable areas of their negotiated specification-compliant reachable sets.

Figure 2.5: General procedure for obtaining specification-compliant trajectories of cooperative vehicles using reachable sets. In this scenario, three cooperative vehicles drive on a three-lane road regulated by a traffic light and with line markings. We show the drivable areas of vehicles disregarding and considering relevant traffic rules, i.e., not crossing solid lines and respecting traffic lights, in subfigures b and c, respectively. In subfigure d, each vehicle receives its negotiated and specification-compliant reachable sets, within which trajectories are individually planned.

Chapter 3
Conclusions

This dissertation covered novel solutions to the specification-compliant motion planning problems for individual and cooperative automated vehicles. In the following sections, we summarize the works of the author included in this dissertation and subsequently discuss possible directions for further improvements and future research.

3.1 Summary

This dissertation covered several methods for individual and cooperative specification-compliant motion planning of automated vehicles using reachability analysis. The core idea involves computing the collision-free reachable sets of automated vehicles, identifying their specification-compliant subsets, and extracting motion planning constraints.

As a first step toward specification-compliant motion planning, we extended the states in our reachable set computation with atomic propositions (see Sec. 2.2) relevant to German traffic rules described in [21]. We successfully identified subsets of our reachable sets that comply with specifications expressed in time-labeled propositional logic, within which trajectories adhering to the same specifications are planned (see [1]).

Considering that traffic rules and other high-level handcrafted rules can be effectively and naturally expressed in temporal logic, as demonstrated in works [18], [19], [21], [22], [27], we coupled reachability analysis with model checking and generated collision-free driving corridors within the reachable sets complying with German traffic rules expressed in metric temporal logic (see [2]). Applying constraints extracted from such driving corridors to motion planners expedites the generation of trajectories complying with enforced temporal specifications. This claim was substantiated by integrating the driving corridors into two sampling-based motion planners and comparing the time and effort required to generate a trajectory adhering to the specifications. We also verified and demonstrated the favorable characteristic of our reachable set computation that it requires less time and effort under more critical scenarios with a smaller solution space. Furthermore, we compared our approach to related work in terms of model accuracy and verification efficiency, exhibiting the superiority of our approach.

In addition to individual motion planning, we extended the results of [1] to a group of cooperative vehicles (see [3]): We computed the specification-compliant reachable sets of each cooperative vehicle and resolved potential conflicts of their reachable sets within the group. As a result, each vehicle receives its own planning space to generate specification-compliant trajectories. The seamless transition from individual to cooperative motion planning considering specifications is demonstrated using examples featuring a precise overtaking situation, a highway scenario, and a roundabout scenario.

Ensuring the safety of automated vehicles at all times can be considered one of the most critical specifications. We extended the concept of the so-called invariably safe sets for individual motion planning [83] to cooperative motion planning (see [4]). This is achieved by incorporating reachability analysis and devising a mechanism for distributing the invariably safe sets to cooperative vehicles. Consequently, the safety of cooperative vehicles can be ensured for an infinite time horizon even under safety-critical situations, e.g., when the leading vehicles perform emergency braking maneuvers.

As a contribution to the scientific community, we released *CommonRoad-Reach*, an open-source toolbox dedicated to the computation of reachable sets and the extraction of driving corridors for automated vehicles (see [5]). Apart from its primary function in trajectory planning for automated vehicles, *CommonRoad-Reach* is versatile and advantageous for a variety of other applications. These include but are not limited to, verifying the absence of collisions between vehicles and estimating the boundaries of future maneuvers of surrounding vehicles. We used various traffic scenarios with different levels of complexity to showcase the functionality of our toolbox. Furthermore, we demonstrated the real-time performance of the toolbox through benchmarking against multiple scenarios from the CommonRoad benchmark suite.

With the primary objectives of testing, verifying, and comparing different motion planners, we developed a pipeline that automatically generates a large number (theoretically as many as allowed by the computational resources) of safety-critical traffic scenarios based on reachability analysis (see [6]). The usefulness of the pipeline has been demonstrated through practical examples such as its employment in the international CommonRoad motion planning competition (years 2021 and 2022).

3.2 Future Work

The preliminary outcomes obtained in this dissertation demonstrate the potential of our proposed methods in generating specification-compliant trajectories for both individual and cooperative automated vehicles, across arbitrary and complex traffic scenarios, without compromising real-time capability. Nevertheless, there are many possible directions worthy of further investigation, some of which are outlined below:

- The computational efficiency of our reachable set computation can be enhanced in several ways. Two possible strategies are described here: (a) Given a set of specifications, at each step, the states in our reachable set computation are extended with atomic propositions pertinent to the specifications. However, not all atomic propositions at every step are essential for determining the truth of the enforced specifications. We suggest analyzing the structure of the specifications, potentially employing automata theory, to exclusively compute the relevant atomic propositions at each step, thereby reducing unnecessary computational efforts and improving the efficiency of reachable set computation; (b) In our current implementation, the reachable set computation does not utilize results from the previous planning cycles. We propose reusing these results as an initial approximation of the reachable sets of the current planning cycle and refining the computation as time permits. This modification ensures the availability of reachable sets at any time of query.

- We recommend investigating novel utility functions to determine the optimal driving corridor among a set of specification-compliant candidates. We present three potential measures: (a) Drivability: Prior to trajectory planning, it is desirable and advantageous to assess if a feasible trajectory exists within a driving corridor to avoid unnecessary computational efforts. (b) Robustness: Although all candidate driving corridors already adhere to the enforced specifications, their degrees of compliance, i.e., robustness degree, may vary. It is preferable to select the driving corridor that satisfies the specifications to a greater extent. (c) Criticality: While all candidate driving corridors are collision-free (with an under-approximative set of forbidden states), their criticality measures, such as the risk of collision, may differ. We could evaluate these measures, for example, using the CommonRoad CriMe [84] toolbox and prioritize safer candidates.

- One possible research direction is the formation of cooperative groups using reachability analysis: Our proposed method for cooperative motion planning, along with most existing literature on the topic, assumes that a cooperative group is established in advance and takes it for granted. However, real-time interactive formation and dissolution of such groups remain open questions. Vehicles should be able to determine when it is advantageous to join or leave a cooperative group, particularly during highly interactive situations, such as passing through larger intersections. We suggest extending the group formation concept of [85] using reachability analysis: Cooperative vehicles could compute their own specification-compliant reachable sets and those of other vehicles. An overlap in the reachable sets indicates that interaction between the vehicles may be necessary, thus promoting the formation of a cooperative group.

- Our proposed method for cooperative motion planning [3] currently only supports specifications expressed in propositional logic. To accommodate traffic rules expressed in temporal logic, it is advisable to extend our approach described in [2] for single vehicles to cooperative vehicles. This necessitates a new mechanism for negotiating conflicting reachable sets, as their temporal relations are vital for determining the truths of specifications. We propose a possible solution consisting of two steps: (a) Each cooperative vehicle identifies its specification-compliant driving corridors as described in [2] and generates a graph containing the base sets of these driving corridors. (b) Perform negotiations if conflicts arise between the base sets of generated graphs in the position domain. The utility function employed in the negotiation should reflect the past, current, and future impacts of a vehicle losing a conflicting base set on its generated graph. This approach enables the incorporation of temporal requirements from the specifications into the negotiation process.

- So far, our proposed methods have only been tested using numerical experiments in relatively simple traffic scenarios from the CommonRoad benchmark suite. It is beneficial to conduct more extensive experiments and tests using traffic simulators such as SUMO and CARLA to identify and address corner cases and safety-critical situations. Apart from utilizing simulated environments, performing real-world test drives with research vehicles, such as EDGAR (see Fig. 1.2), is essential for the practical verification of our theoretical findings.

By addressing these research directions in the future, the work presented in this dissertation can be further advanced, contributing to the ongoing development of specification-compliant motion planning of automated vehicles and paving the way to safer and more efficient road traffic.

Bibliography

[7] S. Singh, “Critical reasons for crashes investigated in the national motor vehicle crash causation survey,” National Center for Statistics and Analysis, U.S. National Highway Traffic Safety Administration, Tech. Rep., 2018.

[8] World Health Organization, *Global status report on road safety 2018*. 2018.

[9] C. Burger, P. F. Orzechowski, Ö. Ş. Taş, and C. Stiller, “Rating cooperative driving: A scheme for behavior assessment,” in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2017, pp. 1–6.

[10] S. Aradi, “Survey of deep reinforcement learning for motion planning of autonomous vehicles,” *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 740–759, 2020.

[11] L. Claussmann, M. Revilloud, D. Gruyer, and S. Glaser, “A review of motion planning for highway autonomous driving,” *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 5, pp. 1826–1848, 2019.

[12] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, “A survey of motion planning and control techniques for self-driving urban vehicles,” *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, 2016.

[13] D. González, J. Pérez, V. Milanés, and F. Nashashibi, “A review of motion planning techniques for automated vehicles,” *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 1135–1145, 2015.

[14] U. N. E. C. for Europe, *Convention on road traffic*, 1968.

[15] G. Bagschik, T. Menzel, and M. Maurer, “Ontology based scene creation for the development of automated vehicles,” in *Proc. of the IEEE Intell. Veh. Symp.*, 2018, pp. 1813–1820.

[16] M. Buechel, G. Hinz, F. Ruehl, H. Schroth, C. Gyoeri, and A. Knoll, “Ontology-based traffic scene modeling, traffic regulations dependent situational awareness and decision-making for automated vehicles,” in *Proc. of the IEEE Intell. Veh. Symp.*, 2017, pp. 1471–1476.

[17] Q. Zhang, D. K. Hong, Z. Zhang, Q. A. Chen, S. Mahlke, and Z. M. Mao, "A systematic framework to identify violations of scenario-dependent driving rules in autonomous vehicle software," *Proc. of the ACM Meas. Anal. Comput. Syst.*, vol. 5, no. 2, pp. 1–25, 2021.

[18] S. Maierhofer, P. Moosbrugger, and M. Althoff, "Formalization of intersection traffic rules in temporal logic," in *Proc. of the IEEE Intell. Veh. Symp.*, 2022, pp. 1135–1144.

[19] Y. Sun, C. M. Poskitt, J. Sun, Y. Chen, and Z. Yang, "LawBreaker: An approach for specifying traffic laws and fuzzing autonomous vehicles," in *Proc. of the IEEE/ACM Int. Conf. Autom. Software Eng.*, 2022, pp. 1–12.

[20] H. Krasowski and M. Althoff, "Temporal logic formalization of marine traffic rules," in *Proc. of the IEEE Intell. Veh. Symp.*, 2021, pp. 186–192.

[21] S. Maierhofer, A.-K. Rettinger, E. C. Mayer, and M. Althoff, "Formalization of interstate traffic rules in temporal logic," in *Proc. of the IEEE Intell. Veh. Symp.*, 2020, pp. 752–759.

[22] K. Esterle, L. Gressenbuch, and A. Knoll, "Formalizing traffic rules for machine interpretability," in *Proc. of the IEEE Connect. Autom. Veh. Symp.*, 2020, pp. 1–7.

[23] A. Pnueli, "The temporal logic of programs," in *Annu. Symp. Found. of Comput. Sci.*, 1977, pp. 46–57.

[24] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-time Syst.*, vol. 2, no. 4, pp. 255–299, 1990.

[25] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Tech., Model., Anal. Timed Fault-Tolerant Syst.* 2004, pp. 152–166.

[26] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theor. Comput. Sci.*, vol. 410, no. 42, pp. 4262–4291, 2009.

[27] A. Rizaldi, J. Keinholz, M. Huber, J. Feldle, F. Immler, M. Althoff, E. Hilgendorf, and T. Nipkow, "Formalising and monitoring traffic rules for autonomous vehicles in Isabelle/HOL," in *Int. Conf. Integr. Formal Methods*, 2017, pp. 50–66.

[28] K. Esterle, V. Aravantinos, and A. Knoll, "From specifications to behavior: Maneuver verification in a semantic state space," in *Proc. of the IEEE Intell. Veh. Symp.*, 2019, pp. 2140–2147.

[29] L. Gressenbuch and M. Althoff, "Predictive monitoring of traffic rules," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2021, pp. 915–922.

[30] M. Hekmatnejad, S. Yaghoubi, A. Dokhanchi, H. B. Amor, A. Shrivastava, L. Karam, and G. Fainekos, "Encoding and monitoring responsibility sensitive safety rules for automated vehicles in signal temporal logic," in *Proc. of the ACM/IEEE Int. Conf. Formal Method. Model. Syst. Des.*, 2019, pp. 1–11.

[31] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *arXiv preprint arXiv:1708.06374*, 2017.

[32] N. Aréchiga, "Specifying safety of autonomous vehicles in signal temporal logic," in *Proc. of the IEEE Intell. Veh. Symp.*, 2019, pp. 58–63.

[33] H. Roehm, J. Oehlerking, T. Heinz, and M. Althoff, "STL model checking of continuous and hybrid systems," in *Int. Symp. Autom. Technol. Verif. Anal.*, 2016, pp. 412–427.

[34] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *Int. J. Rob. Res.*, vol. 20, no. 5, pp. 378–400, 2001.

[35] R. R. Da Silva, V. Kurtz, and H. Lin, "Automatic trajectory synthesis for real-time temporal logic," *IEEE Trans. Autom. Control*, vol. 67, no. 2, pp. 780–794, 2021.

[36] C. K. Verginis, C. Vrohidis, C. P. Bechlioulis, K. J. Kyriakopoulos, and D. V. Dimarogonas, "Reconfigurable motion planning and control in obstacle cluttered environments under timed temporal tasks," in *Proc. of the IEEE Int. Conf. Robot. Autom.*, 2019, pp. 951–957.

[37] Y. Shoukry, P. Nuzzo, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada, "SMC: Satisfiability modulo convex programming," *Proc. of the IEEE*, vol. 106, no. 9, pp. 1655–1679, 2018.

[38] K. Cho, J. Suh, C. J. Tomlin, and S. Oh, "Cost-aware path planning under co-safe temporal logic specifications," *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 2308–2315, 2017.

[39] M. Lahijanian, M. R. Maly, D. Fried, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi, "Iterative temporal planning in uncertain environments with partial satisfaction guarantees," *IEEE Trans. Rob.*, vol. 32, no. 3, pp. 583–599, 2016.

[40] Y. Zhou, D. Maity, and J. S. Baras, "Timed automata approach for motion planning using metric interval temporal logic," in *Proc. of the Eur. Control Conf.*, 2016, pp. 690–695.

[41] D. Maity and J. S. Baras, "Motion planning in dynamic environments with bounded time temporal logic specifications," in *Mediterr. Conf. Control Autom.*, 2015, pp. 940–946.

[42] R. Kohlhaas, T. Bittner, T. Schamm, and J. M. Zöllner, "Semantic state space for high-level maneuver planning in structured traffic scenes," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2014, pp. 1060–1065.

[43] E. M. Wolff, U. Topcu, and R. M. Murray, "Automaton-guided controller synthesis for nonlinear systems with temporal logic," in *Proc. of the IEEE Int. Conf. Intell. Robot. Syst.*, 2013, pp. 4332–4339.

[44] M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, *50 years of integer programming 1958-2008: From the early years to the state-of-the-art.* Springer Science & Business Media, 2009.

[45] D. Sun, J. Chen, S. Mitra, and C. Fan, "Multi-agent motion planning from signal temporal logic specifications," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 3451–3458, 2022.

[46] Z. Lin and J. S. Baras, "Optimization-based motion planning and runtime monitoring for robotic agent with space and time tolerances," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1874–1879, 2020.

[47] U. A. Fiaz and J. S. Baras, "Fast, composable rescue mission planning for UAVs using metric temporal logic," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 15 404–15 411, 2020.

[48] S. Saha and A. A. Julius, "An MILP approach for real-time optimal controller synthesis with metric temporal logic specifications," in *Proc. of the Am. Control Conf.*, 2016, pp. 1105–1110.

[49] E. M. Wolff and R. M. Murray, "Optimal control of nonlinear systems with temporal logic specifications," in *Rob. Res.* 2016, pp. 21–37.

[50] C. I. Vasile, X. Li, and C. Belta, "Reactive sampling-based path planning with temporal logic specifications," *Int. J. Rob. Res.*, vol. 39, no. 8, pp. 1002–1028, 2020.

[51] J. Karlsson, F. S. Barbosa, and J. Tumova, "Sampling-based motion planning with temporal logic missions and spatial preferences," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 15 537–15 543, 2020.

[52] F. S. Barbosa, L. Lindemann, D. V. Dimarogonas, and J. Tumova, "Integrated motion planning and control under metric interval temporal logic specifications," in *Eur. Control Conf.*, 2019, pp. 2042–2049.

[53] C.-I. Vasile, V. Raman, and S. Karaman, "Sampling-based synthesis of maximally-satisfying controllers for temporal logic specifications," in *Proc. of the IEEE Int. Conf. Intell. Robot. Syst.*, 2017, pp. 3840–3847.

[54] L. I. R. Castro, P. Chaudhari, J. Tůmová, S. Karaman, E. Frazzoli, and D. Rus, "Incremental sampling-based algorithm for minimum-violation motion planning," in *Proc. of the IEEE Conf. Decis. Control*, IEEE, 2013, pp. 3217–3224.

[55] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Rob. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[56] L. Zhang and D. Manocha, "An efficient retraction-based RRT planner," in *Proc. of the IEEE Int. Conf. Robot. Autom.*, IEEE, 2008, pp. 3743–3750.

[57] S. Malik, M. A. Khan, and H. El-Sayed, "Collaborative autonomous driving — a survey of solution approaches and future challenges," *Sensors*, vol. 21, no. 11, pp. 3783–3806, 2021.

[58] B. Häfner, V. Bajpai, J. Ott, and G. A. Schmitt, "A survey on cooperative architectures and maneuvers for connected and automated vehicles," *IEEE Commun. Surv. Tutorials*, vol. 24, no. 1, pp. 380–403, 2021.

[59] U. Montanaro, S. Dixit, S. Fallah, M. Dianati, A. Stevens, D. Oxtoby, and A. Mouzakitis, “Towards connected autonomous driving: Review of use-cases,” *Veh. Syst. Dyn.*, vol. 57, no. 6, pp. 779–814, 2019.

[60] Z. Wang, Y. Bian, S. E. Shladover, G. Wu, S. E. Li, and M. J. Barth, “A survey on cooperative longitudinal motion control of multiple connected and automated vehicles,” *IEEE Intell. Intell. Transp. Syst. Mag.*, vol. 12, no. 1, pp. 4–24, 2019.

[61] J. Rios-Torres and A. A. Malikopoulos, “A survey on the coordination of connected and automated vehicles at intersections and merging at highway on-ramps,” *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 5, pp. 1066–1077, 2017.

[62] K. Dresner and P. Stone, “Multiagent traffic management: A reservation-based intersection control mechanism,” in *Proc. of the Int. Joint Conf. Auton. Agents and Multiagent Syst.*, 2004, pp. 530–537.

[63] S. Manzinger and M. Althoff, “Tactical decision making for cooperative vehicles using reachable sets,” in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2018, pp. 444–451.

[64] F. Zhu and S. V. Ukkusuri, “A linear programming formulation for autonomous intersection control within a dynamic traffic assignment and connected vehicle environment,” *Transp. Res. Part C: Emerg. Technol.*, vol. 55, pp. 363–378, 2015.

[65] M. W. Levin, H. Fritz, and S. D. Boyles, “On optimizing reservation-based intersection controls,” *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 3, pp. 505–515, 2017.

[66] D. Heß, R. Lattarulo, J. Pérez, J. Schindler, T. Hesse, and F. Köster, “Fast maneuver planning for cooperative automated vehicles,” in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2018, pp. 1625–1632.

[67] M. Nichting, D. Heß, J. Schindler, T. Hesse, and F. Köster, “Space time reservation procedure (STRP) for V2X-based maneuver coordination of cooperative automated vehicles in diverse conflict scenarios,” in *Proc. of the IEEE Intell. Veh. Symp.*, 2020, pp. 502–509.

[68] D. Heß, R. Lattarulo, J. Pérez, T. Hesse, and F. Köster, “Negotiation of cooperative maneuvers for automated vehicles: Experimental results,” in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2019, pp. 1545–1551.

[69] M. Vasirani and S. Ossowski, “A market-inspired approach for intersection management in urban road traffic networks,” *J. Artif. Intell. Res.*, vol. 43, pp. 621–659, 2012.

[70] H. Rewald and O. Stursberg, “Cooperation of autonomous vehicles using a hierarchy of auction-based and model-predictive control,” in *Proc. of the IEEE Intell. Veh. Symp.*, 2016, pp. 1078–1084.

[71] P. Bender, J. Ziegler, and C. Stiller, “Lanelets: Efficient map representation for autonomous driving,” in *Proc. of the IEEE Intell. Veh. Symp.*, 2014, pp. 420–425.

[72] G. De Giacomo, A. Murano, F. Patrizi, and G. Perelli, "Timed trace alignment with metric temporal logic over finite traces," in *Proc. of the Int. Conf. Principles Knowl. Represent. and Reasoning*, vol. 18, 2021, pp. 227–236.

[73] D. D'Souza and P. Prabhakar, "On the expressiveness of MTL in the pointwise and continuous semantics," *Int. J. Software Tools for Technol. Transfer*, vol. 9, no. 1, pp. 1–4, 2007.

[74] A. Cecconi, C. D. Ciccio, G. D. Giacomo, and J. Mendling, "Interestingness of traces in declarative process mining: The Janus LTLp$_f$ approach," in *Int. Conf. Bus. Process Manage.*, Springer, 2018, pp. 121–138.

[75] M. Huth and M. Ryan, *Logic in computer science: Modelling and reasoning about systems.* Cambridge university press, 2004.

[76] G. Lafferriere, G. J. Pappas, and S. Yovine, "Symbolic reachability computation for families of linear vector fields," *J. Symb. Comput.*, vol. 32, no. 3, pp. 231–253, 2001.

[77] J. Eilbrecht and O. Stursberg, "Challenges of trajectory planning with integrator models on curved roads," in *Proc. of the IFAC World Congr.*, 2020, pp. 15 588–15 595.

[78] S. Söntges and M. Althoff, "Computing the drivable area of autonomous road vehicles in dynamic road scenes," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 6, pp. 1855–1866, 2018.

[79] M. Althoff, G. Frehse, and A. Girard, "Set propagation techniques for reachability analysis," *Annu. Rev. Control Rob. Auton. Syst.*, vol. 4, no. 1, pp. 369–395, 2021.

[80] H. T. Croft, K. Falconer, and R. K. Guy, *Unsolved problems in geometry: Unsolved problems in intuitive mathematics.* Springer Science & Business Media, 2012, vol. 2.

[81] L. Schäfer, S. Manzinger, and M. Althoff, "Computation of solution spaces for optimization-based trajectory planning," *IEEE Trans. Intell. Veh.*, vol. 8, no. 1, pp. 216–231, 2021.

[82] S. Manzinger, C. Pek, and M. Althoff, "Using reachable sets for trajectory planning of automated vehicles," *IEEE Trans. Intell. Veh.*, vol. 6, no. 2, pp. 232–248, 2020.

[83] C. Pek and M. Althoff, "Efficient computation of invariably safe states for motion planning of self-driving vehicles," in *Proc. of the IEEE Int. Conf. Intell. Robot. Syst.*, 2018, pp. 3523–3530.

[84] Y. Lin and M. Althoff, "CommonRoad-CriMe: A toolbox for criticality measures of autonomous vehicles," in *Proc. of the IEEE Intell. Veh. Symp.*, 2023, pp. 1–8.

[85] C. Frese, J. Beyerer, and P. Zimmer, "Cooperation of cars and formation of cooperative groups," in *Proc. of the IEEE Intell. Veh. Symp.*, 2007, pp. 227–232.

Appendix A
Reproduction of Publications

The following statement applies to articles [1], [2], [4]–[6]:

A.1 Computing Specification-Compliant Reachable Sets for Motion Planning of Automated Vehicles [1]

Summary To ensure the seamless integration of automated vehicles into road traffic, there is a pressing need for these vehicles to adhere to specifications, including traffic rules. This article presents a method that incorporates specifications expressed in time-labeled propositional logic into the reachability analysis of automated vehicles, within which trajectories can be planned to meet the same specifications.

Applying reachability analysis on an automated vehicle in an over-approximative manner yields a set of states (*reachable sets*) that encloses all drivable trajectories of the vehicle over time. Depending on the enforced specifications, semantic labels are generated for reachable sets from relevant predicates concerning the vehicle's positions, velocities, accelerations, and general traffic situations. The semantically-annotated reachable sets are checked against specifications, whose compliant subsets serve as low-level motion planning constraints to expedite the generation of trajectories complying with the same specifications. In addition to the benefits of employing reachability analysis for motion planning, the distinctive characteristics of our method include its capacity to (1) expedite the adherence to traffic rules formalized in [21], (2) effectively integrate handcrafted specifications using the predicates mentioned above, and (3) be integrated with motion planners accepting position and velocity constraints. Using traffic scenarios from the CommonRoad benchmark suite, we exhibit the applicability of the proposed method for obtaining specification-compliant trajectories.

Contributions of E. I. L. E. I. L. developed the idea of the research (together with M. A.); E. I. L. designed, conducted, and evaluated the experiments; E. I. L. wrote the article (together with M. A.).

Conference article The accepted version of the article is reprinted. The final version of record is available at `https://doi.org/10.1109/IV48863.2021.9575739`.

Attachment The animation of the evaluations is available at `https://mediatum.ub.tum.de/1595757`.

Computing Specification-Compliant Reachable Sets for Motion Planning of Automated Vehicles

Edmond Irani Liu and Matthias Althoff

***Abstract*— To safely and effectively participate in road traffic, automated vehicles should explicitly consider compliance with traffic rules and high-level specifications. We propose a method that can incorporate traffic and handcrafted rules expressed in time-labeled propositional logic into our reachability analysis, which computes the over-approximative set of states reachable by vehicles. These reachable sets serve as low-level trajectory planning constraints to expedite the search for specification-compliant trajectories. Depending on the adopted specifications, related semantic labels are generated from predicates considering positions, velocities, accelerations, and general traffic situations. We exhibit the applicability of the proposed method with scenarios from the CommonRoad benchmark suite.**

I. Introduction

Highly automated vehicles (AVs) promise increased road safety compared with human-driven ones. To safely and effectively participate in road traffic, AVs should explicitly consider compliance with traffic and handcrafted rules. Compliance with the former exempts manufacturers from potential liability claims in case an accident happens, whereas the latter contribute to finding motion plans that meet specific requirements.

Determining a drivable trajectory that satisfies a desired discrete specification involves reasoning with both discrete and continuous states of AV, which poses computational challenges originating from (a) vehicle dynamics and collision avoidance, (b) discrete specifications, and (c) interwoven dependencies between continuous trajectories and discrete constraints. Planning on the discrete level may output plans that meet the specifications but do not satisfy dynamic constraints; similarly, motion planning methods may generate collision-free and dynamically feasible trajectories that violate the specifications.

In this study, we address these challenges by extending our previous work [1] to compute specification-compliant reachable sets for a considered ego vehicle. Our over-approximative reachable sets enclose all drivable trajectories of AV and can be used as low-level trajectory planning constraints [2], which expedite the search for specification-compliant trajectories. Depending on the adopted specifications expressed in time-labeled propositional logic, relevant semantic labels are attached to the reachable sets. An exemplary handcrafted rule can be described as "follow vehicle A up to time step k_1, then finish overtaking it from the left before time step k_2."

All authors are with the Department of Informatics, Technical University of Munich, 85748 Garching, Germany.
{edmond.irani,althoff}@tum.de

The efforts to obtain a specification-compliant trajectory can be roughly categorized into three groups. The first group generates discrete plans to guide the trajectory planning process. For example, Shoukry *et al.* [3] proposed a satisfiability modulo convex programming framework to handle Boolean and convex constraints; Lahijanian *et al.* [4] proposed a multilayered synergistic framework, which is an extension of [5] to cope with newly discovered obstacles. Zhou *et al.* [6] used timed automata to synthesize timed paths that satisfy considered specifications. In these methods, the high-level planners suggest discrete plans based on an abstraction of a system, and the low-level motion planners generate trajectories that comply with the discrete plans. Since the continuous constraints are not explicitly considered by the high-level planners, the drivability of the suggested plans is often not ensured [7], thereby requiring frequent replanning on the discrete level. The second group evaluates the specifications on the planned trajectories with monitors [8], [9]. The monitoring can be efficiently performed; however, in case the trajectory under examination is rejected by the monitor, no alternative candidate trajectory is returned. The third group, to which the present study belongs, considers the specifications before trajectory planning, e.g., in high-level maneuver planners [10]–[13]. Kohlhaas *et al.* [13] generated maneuvers with simple traffic rules by traversing a graph represented in a semantic state space; Esterle *et al.* [10] adopted a similar idea and generated maneuvers that complied with linear temporal logic (LTL) specifications. In a previous work [12], we output so-called driving corridors that satisfied sequences of position relations to other vehicles.

Contrary to these works, we not only consider specifications with position relations to other obstacles but also include predicates considering velocities, accelerations, and general traffic situations based on the recent formalization of German road traffic regulations in temporal logic [14], [15]. In [14], traffic rules related to velocities, overtaking, safe distances, priorities, etc, were formalized using LTL. In comparison, metric temporal logic (MTL), which is an extension of LTL to support time intervals, was used in [15] to formalize a selection of traffic rules related to general and interstate driving situations. The proposed method stands out in that it

1) integrates predicates into reachable set computation for compliance with traffic rules formalized in [15];
2) can incorporate handcrafted specifications with the above-mentioned predicates; and
3) can be combined with any motion planning algorithms.

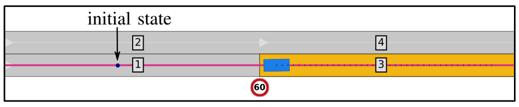


Fig. 1: A scenario with four lanelets (with IDs 1–4 shown with numbered boxes) and a leading vehicle (blue). Lanelet 3 is the goal lanelet (yellow) of an ego vehicle. The straight line (magenta) indicates a reference path leading to the goal lanelet.

The remainder of this article is organized as follows: Sec. II introduces basic definitions and necessary preliminaries. Subsequently, we present the partitioning of the state space and the computation of reachable sets in Sec. III. Afterward, we demonstrate evaluations with exemplary scenarios from the CommonRoad[1] benchmark suite [16] in Sec. IV. Finally, we conclude in Sec. V.

II. Preliminaries

A. System Description

In this study, the considered scenarios are described in the CommonRoad format. A typical CommonRoad scenario (see Fig. 1) consists of (a) a road network represented by lanelets [17], whose left and right bounds are modeled with polylines, (b) static and dynamic obstacles, and (c) traffic rule elements, such as traffic signs, traffic lights, and road markings. We use the trajectories provided in the scenarios and consider them as adequate prediction for obstacles over time. Alternatively, one can adopt a set-based prediction [18] for obstacles. Given a planning problem, which includes the initial state of an ego vehicle and a set of goal states, a reference path to the goal lanelet is planned. This path is used to construct the local curvilinear coordinate system of the ego vehicle, as described in [17]. The choice of this coordinate system facilitates the formulation of maneuvers from the ego vehicle's perspective, e.g., lane-following, stopping before an intersection, and preventing driving backward.

The system dynamics of the ego vehicle is abstracted by a point-mass model with the center of the vehicle as the reference point. The model is represented with two second-order integrators in longitudinal s-direction and lateral d-direction of the curvilinear coordinate system. Let $\square$ be a variable, we denote by $\underline{\square}$ and $\overline{\square}$ its minimum and maximum values, respectively. In addition, we attach subscripts s/d to variables to indicate the directions in which they are described. The system dynamics of the ego vehicle is

$$x_{k+1} = \begin{pmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{pmatrix} x_k + \begin{pmatrix} \frac{1}{2}\Delta t^2 & 0 \\ \Delta t & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ 0 & \Delta t \end{pmatrix} u_k, \quad (1)$$

where $k \in \mathbb{N}_0$ is a discrete time step corresponding to a point in time $t_k = k\Delta t$, with $\Delta t \in \mathbb{R}_+$ being a predefined time increment. $x \in \mathcal{X} \subset \mathbb{R}^4$ is a state in the state space $\mathcal{X}$, and $u \in \mathcal{U} \subset \mathbb{R}^2$ is an input. The state of the vehicle is modeled as $x = (p_s, v_s, p_d, v_d)^\mathsf{T}$, with p and v representing the position and velocity, respectively. The system accepts inputs $u = (a_s, a_d)^\mathsf{T}$, where a is the acceleration. The velocities and accelerations in both directions are bounded by the over-approximation of the physically feasible values:

$$\underline{v}_s \leq v_{s,k} \leq \overline{v}_s, \ \ \underline{v}_d \leq v_{d,k} \leq \overline{v}_d, \quad (2a)$$

$$\underline{a}_s \leq a_{s,k} \leq \overline{a}_s, \ \ \underline{a}_d \leq a_{d,k} \leq \overline{a}_d. \quad (2b)$$

The bounds are chosen conservatively to consider the kinematic limitations and effects arising from transforming the system dynamics to the curvilinear coordinate system. Notably, the drivability of the planned trajectories should be examined individually, e.g., using the drivability checker described in [19]. Finally, we denote the length of the ego vehicle by l.

B. Reachable Sets

Given an initial state x_0 and an input trajectory $u_{[0,k]}$, we use $\chi_k(x_0, u_{[0,k]})$ to represent the solution to (1) at time step k. We assume a set of time-dependent obstacles to be given, the union of whose occupancies at time step k is represented by $\mathcal{O}_k \subset \mathbb{R}^2$. The sets of states whose occupancy (considering the shape of the ego vehicle) are overlapping with $\mathcal{O}_k$ are removed from the reachable sets. Let $\mathcal{X}_k^{\mathsf{CF}} = \mathcal{X} \setminus \mathcal{O}_k$ be the set of collision-free states at time step k, the exact collision-free reachable set of the ego vehicle at k starting from the initial set of states $\mathcal{X}_0$ is

$$\mathcal{R}_k^*(\mathcal{X}_0) := \Big\{ \chi_k(x_0, u_{[0,k]}) \Big| x_0 \in \mathcal{X}_0, \forall \tau \in \{0, \dots, k\} : u_\tau \in \mathcal{U}, \chi_\tau(x_0, u_{[0,\tau]}) \in \mathcal{X}_\tau^{\mathsf{CF}} \Big\}.$$

Subsequently, we omit $\mathcal{X}_0$ for convenience. Obtaining $\mathcal{R}_k^*$ is computationally demanding in general; therefore, we instead compute its over-approximation $\mathcal{R}_k$, which is the union of so-called base sets $\mathcal{R}_k^{(i)}$, $i \in \mathbb{N}$. Each base set $\mathcal{R}_k^{(i)} = \hat{\mathcal{P}}_{s,k}^{(i)} \times \hat{\mathcal{P}}_{d,k}^{(i)}$ is chosen to be a Cartesian product of two convex polytopes $\hat{\mathcal{P}}_{s,k}^{(i)}$ and $\hat{\mathcal{P}}_{d,k}^{(i)}$ which represent the reachable positions and velocities in (p_s, v_s) and (p_d, v_d) planes, respectively (see Fig. 2a–b) [1]. This choice is motivated by the existence of efficient algorithms for required set operations on convex polytopes. To simplify the notation, we also denote the collection of $\mathcal{R}_k^{(i)}$ by $\mathcal{R}_k$, i.e., $\mathcal{R}_k = \{\mathcal{R}_k^{(1)}, \dots, \mathcal{R}_k^{(i)}\}$. The projection of $\mathcal{R}_k^{(i)}$ onto the position domain yields axis-aligned rectangles $\mathcal{D}_k^{(i)}$ (see Fig. 2c), whose union is referred to as the drivable area $\mathcal{D}_k$. Similarly, we use $\mathcal{D}_k$ to denote the collection of $\mathcal{D}_k^{(i)}$.

Definition 1 (Projection):
The operator $\mathrm{proj}_\diamondsuit(\cdot)$ *maps the input to its elements* $\diamondsuit$*. For example,* $\mathrm{proj}_{(p,v)}(x) = (p, v)^\mathsf{T}$ *for* $x = (p, v, a)^\mathsf{T}$*. A set* $\mathcal{X}$ *can be projected using the same notation:* $\mathrm{proj}_\diamondsuit(\mathcal{X}) = \{\mathrm{proj}_\diamondsuit(x) | x \in \mathcal{X}\}$.

Definition 2 (Drivable Area):
The drivable area is defined as the projection of the reachable set onto the position domain: $\mathcal{D}_k := \mathrm{proj}_{(p_s, p_d)}(\mathcal{R}_k)$.

[1]https://commonroad.in.tum.de/

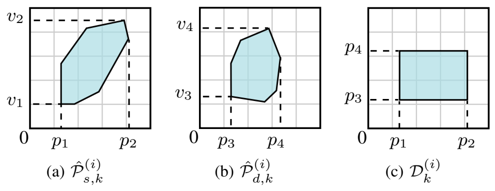


Fig. 2: Polytopes and drivable area of a base set $\mathcal{R}_k^{(i)}$.

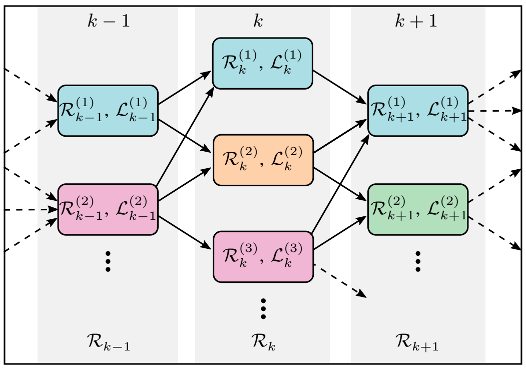

$k-1$
k
$k+1$
$\mathcal{R}_k^{(1)}, \mathcal{L}_k^{(1)}$
$\mathcal{R}_{k-1}^{(1)}, \mathcal{L}_{k-1}^{(1)}$
$\mathcal{R}_{k+1}^{(1)}, \mathcal{L}_{k+1}^{(1)}$
$\mathcal{R}_k^{(2)}, \mathcal{L}_k^{(2)}$
$\mathcal{R}_{k-1}^{(2)}, \mathcal{L}_{k-1}^{(2)}$
$\mathcal{R}_{k+1}^{(2)}, \mathcal{L}_{k+1}^{(2)}$
$\mathcal{R}_k^{(3)}, \mathcal{L}_k^{(3)}$
$\mathcal{R}_{k-1}$
$\mathcal{R}_k$
$\mathcal{R}_{k+1}$

Fig. 3: Reachability graph $\mathcal{G}_\mathcal{R}$ holding nodes of different time steps. The nodes with the same labels have the same color.

In this study, each base set $\mathcal{R}_k^{(i)}$ additionally carries a set $\mathcal{L}_k^{(i)}$ of semantic labels, whose collection is denoted by $\mathcal{L}_k$. The generation of these labels will be explained in Sec. III-D.3. Let us introduce the reachability graph $\mathcal{G}_\mathcal{R}$, which is a directed graph connecting base sets $\mathcal{R}_k^{(i)}$ so that their temporal and spatial relationships can be queried (see Fig. 3). In $\mathcal{G}_\mathcal{R}$, each node represents exactly one base set $\mathcal{R}_k^{(i)}$. An edge connecting $\mathcal{R}_k^{(i)}$ and $\mathcal{R}_{k+1}^{(j)}$ indicates that $\mathcal{R}_{k+1}^{(j)}$ is reachable from $\mathcal{R}_k^{(i)}$. A base set $\mathcal{R}_k^{(i)}$ may reach several base sets $\mathcal{R}_{k+1}^{(j)}$ in the next time step.

III. Methodology

To obtain specification-compliant reachable sets, we should (a) semantically label reachable sets with relevant predicates and (b) constrain reachable sets to subsets satisfying the desired specifications. We partition the state space based on position predicates to expedite the labeling process. Similar strategies have been employed in [10], [13], [20]. We do not consider velocity predicates at this stage since it requires the computationally demanding splitting of the state space with (non)linear curves (see Fig. 5c–d). Instead, we directly evaluate them on individual reachable sets (detailed in Sec. III-D.2). The selection of considered predicates is listed in Tab. I: Evaluating a *dynamic* predicate is obstacle-dependent, whereas that of a *static* predicate is not.

A. Formulation of Partitions

Computing the partitions of the state space involves operations such as set intersection and difference. To avoid gross approximations while maintaining the computational complexity at an acceptable level, they are modeled with a set of hyperrectangles $\texttt{rect}_q$. Notably, such a choice is not mandatory, any other representation that captures the partitions suffices. Each $\texttt{rect}_q$ is defined as a Cartesian product of intervals over the position and velocity domains:

$$\begin{aligned}\texttt{rect}_q := &\big([\underline{p}_{q,s}, \overline{p}_{q,s}] \times [\underline{v}_{q,s}, \overline{v}_{q,s}]\big)\times \\ &\big([\underline{p}_{q,d}, \overline{p}_{q,d}] \times [\underline{v}_{q,d}, \overline{v}_{q,d}]\big),\end{aligned} \tag{3}$$

where $p_{q,s}$ and $v_{q,s}$ denote the position and velocity of the q-th hyperrectangle in the s-direction, respectively. The same applies to $p_{q,d}$ and $v_{q,d}$ in the d-direction. A regular grid of pairwise-disjoint axis-aligned cells is formed along the reference path. Let $\mathcal{C}^{(q)} = [\underline{p}_{q,s}, \overline{p}_{q,s}] \times [\underline{p}_{q,d}, \overline{p}_{q,d}] \subset \mathbb{R}^2$ be the q-th cell in the grid. By computing the Cartesian product of $\mathcal{C}^{(q)}$ and velocity intervals $[\underline{v}_{q,s}, \overline{v}_{q,s}]$ and $[\underline{v}_{q,d}, \overline{v}_{q,d}]$, a hyperrectangle $\texttt{rect}_q$ can be created. By default, the velocity intervals $[\underline{v}_s, \overline{v}_s]$ and $[\underline{v}_d, \overline{v}_d]$ (see (2a)) are used.

Let $\mathcal{P} = \{\texttt{pred}_1, \texttt{pred}_2, \dots\}$ be the set of considered position predicates, with its power set denoted by $2^\mathcal{P}$. We denote by $\mathrm{part}(k; \mathcal{Z}_j)$ the set of hyperrectangles for which the predicates in $\mathcal{Z}_j \in 2^\mathcal{P}$ evaluate to $\texttt{True}$ at time step $k \in \mathbb{N}_0$. $\mathcal{Z}_j$ is realizable if $\exists k \in \{0, \dots, k_h\} : \mathrm{part}(k; \mathcal{Z}_j) \neq \varnothing$, with k_h being a predefined planning horizon. Fig. 5b illustrates exemplary partitions projected onto the (p_s, v_s) plane. We aim to obtain the collection $\mathcal{Z} \subseteq 2^\mathcal{P}$ of all realizable $\mathcal{Z}_j$ considering relevant lanelets and obstacles:

$$\mathcal{Z} = \Big\{\mathcal{Z}_j \in 2^\mathcal{P} \Big| \exists k \in \{0, \dots, k_h\} : \mathrm{part}(k; \mathcal{Z}_j\}) \neq \varnothing\Big\}.$$

$\mathcal{Z}$ is used for splitting of reachable sets (see Sec. III-D.2).

B. Evaluation of Position Predicates

1) Static Position Predicates: These predicates do not depend on obstacles. We formulate two examples as follows:

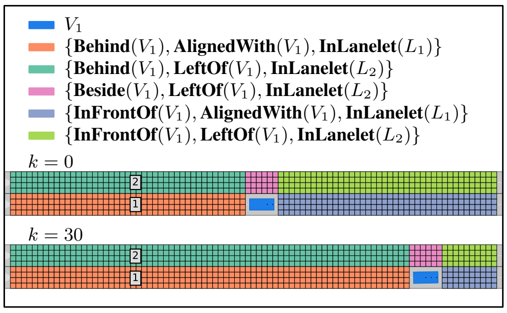
- **InLanelet**$(k; \texttt{rect}_q, L_\texttt{id})$: $\texttt{True}$ if $\texttt{rect}_q$ is within the lanelet with ID $\texttt{id}$, denoted by $L_\texttt{id}$, at time step k.
- **DrivesRightmost**$(k; \texttt{rect}_q, \texttt{area})$: $\texttt{True}$ if $\texttt{rect}_q$ intersects with the rightmost area of lanelets, denoted by $\texttt{area}$. The distance between any point within this area to the right bound of the lanelet does not exceed a predefined distance.

We use $\mathcal{Z}^\mathtt{L} \subseteq 2^\mathcal{P}$ to denote the power set of considered static position predicates. For the sake of brevity, we only keep the lanelets and obstacles (explained later) in the arguments of predicates in the rest of this work.

2) Dynamic Position Predicates: These predicates reflect position relationships between the ego vehicle and obstacles. In this study, we use vehicles as examples of obstacles. Let $\mathcal{V} = \{V_1, \dots, V_N\}$ be the set of other vehicles with IDs $\mathcal{N} = \{1, \dots, N\}$. In addition, let $\mathrm{occ}(k; V_n)$ return the occupancy of V_n at time step k, with bounds in the s-direction denoted by $\underline{p}_{n,s,k}$ and $\overline{p}_{n,s,k}$, respectively. Along the s-direction, the mutually exclusive predicates $\mathcal{P}_{n,s} = \{\{\textbf{InFrontOf}(V_n)\}, \{\textbf{Behind}(V_n)\}, \{\textbf{Beside}(V_n)\}\}$ can be evaluated on $\texttt{rect}_q$ with respect to V_n at time step k as follows:

TABLE I: The selection of considered predicates inspired by [15].

Category	Type	Predicate	Source/Inspiration
Position	Static	**InLanelet**, **DrivesRightmost**, **OnMainCarriageWay**, **OnAccessRamp**, . . .	R_I2, R_I4
	Dynamic	**Behind**, **Beside**, **InFrontOf**, **LeftOf**, **AlignedWith**, **RightOf**, . . .	R_G1, R_I2
Velocity	Static	**BelowFOVVLimit**, **BelowTypeVLimit**, **AboveMinimumVLimit**, . . .	R_G3, R_I1
	Dynamic	**SafeFollowingVelocity(To)**, **SafeLeadingVelocity(To)**, **DrivesFaster**, . . .	R_G1, R_I2
Acceleration	Static	**AdmissibleBraking**, . . .	R_G2
General	Static	**ChangeLanelet**, **PreservesTrafficFlow**, **StandingStill**, . . .	R_G4, R_I1
	Dynamic	**InCongestion**, **SlowLeadingVehicle**, . . .	R_G1, R_G4, R_I1



🟦 V_1
🟧 $\{$**Behind**(V_1), **AlignedWith**(V_1), **InLanelet**$(L_1)\}$
🟩 $\{$**Behind**(V_1), **LeftOf**(V_1), **InLanelet**$(L_2)\}$
🟪 $\{$**Beside**(V_1), **LeftOf**(V_1), **InLanelet**$(L_2)\}$
▨ $\{$**InFrontOf**(V_1), **AlignedWith**(V_1), **InLanelet**$(L_1)\}$
🟩 $\{$**InFrontOf**(V_1), **LeftOf**(V_1), **InLanelet**$(L_2)\}$
$k = 0$
2
1
$k = 30$
2
1

Fig. 4: Projection of the partitions of $\mathcal{Z}_j \in \mathcal{Z}$ onto the position domain. Lanelet IDs are shown with numbered boxes.

- **InFrontOf**(V_n): $\texttt{True}$ if $\underline{p}_{q,s} - l/2 > \overline{p}_{n,s,k}$.
- **Behind**(V_n): $\texttt{True}$ if $\overline{p}_{q,s} + l/2 < \underline{p}_{n,s,k}$.
- **Beside**(V_n): $\texttt{True}$ if $\neg$**InFrontOf**$(V_n) \wedge \neg$**Behind**(V_n) $\wedge\, \mathrm{proj}_{(p_s,p_d)}(\texttt{rect}_q) \cap \mathrm{occ}(k; V_n) = \emptyset$.

Analogously, along the d-direction, the mutually exclusive set of predicates $\mathcal{P}_{n,d} = \big\{\{$**RightOf**$(V_n)\}$...

Analogously, along the d-direction, the mutually exclusive set of predicates $\mathcal{P}_{n,d} = \big\{\{$**LeftOf**$(V_n)\}$, $\{$**RightOf**$(V_n)\}$, $\{$**AlignedWith**$(V_n)\}\big\}$ can be evaluated on $\texttt{rect}_q$.

C. Realizable Sets of Position Predicates

The operator $\mathrm{product}(\cdot)$ over $\tilde{n}$ collections $\mathcal{A}_1, \dots, \mathcal{A}_{\tilde{n}}$ is defined as

$$\mathrm{product}(\mathcal{A}_1, \dots, \mathcal{A}_{\tilde{n}}) = \big\{\mathcal{A}'_1 \cup \dots \cup \mathcal{A}'_{\tilde{n}} \big| \; \mathcal{A}'_i \in \mathcal{A}_i, i \in \{1, \dots, \tilde{n}\}\big\}.$$

As an example, in the presence of a vehicle V_1, $\mathrm{product}(\mathcal{P}_{1,s}, \mathcal{P}_{1,d}) = \big\{\{$**InFrontOf**$(V_1)$, **LeftOf**$(V_1)\}, \dots,$ $\{$**Beside**(V_1), **AlignedWith**$(V_1)\}\big\}$ (see Sec. III-B.2). We denote by $\mathcal{Z}^{\mathtt{V}}_n$ the collection of realizable sets of position predicates created with respect to V_n that can be projected onto the curvilinear coordinate system of the ego vehicle. The formulation of $\mathcal{Z}^{\mathtt{V}}_n$ and $\mathcal{Z}$ (see Sec. III-A) are presented in Alg. 1: we iteratively examine all possible combinations of predicates and keep the ones that have a nonempty partition for at least one time step within the planning horizon k_h. Fig. 4 illustrates the projection of the partitions of $\mathcal{Z}_j \in \mathcal{Z}$ created for an exemplary scenario onto the position domain at two time steps. Owing to our formulation of the predicates, the mentioned projection is collision-free with respect to the obstacles and are pairwise-disjoint at any specific time step.

Algorithm 1 Realizable Sets of Position Predicates

Inputs: Set $\mathcal{N}$ of IDs of vehicles, planning horizon k_h, Collection $\mathcal{Z}^{\mathtt{L}}$

Output: Collection $\mathcal{Z}$

```
for $n$ in $\mathcal{N}$ do                                   ▷ Formulation of $\mathcal{Z}^{\mathtt{V}}_n$
    $\mathcal{Z}^{\mathtt{V}}_n \leftarrow \{\}$
    $\mathcal{P}_{n,s}, \mathcal{P}_{n,d} \leftarrow$ OBTAINPREDICATES($n$)    ▷ see Sec. III-B.2
    for $\mathcal{Z}_i$ in product($\mathcal{P}_{n,s}, \mathcal{P}_{n,d}$) do
        if HASNONEMPTYPARTITION($\mathcal{Z}_i$) then
            $\mathcal{Z}^{\mathtt{V}}_n$.ADD($\mathcal{Z}_i$)
        end if
    end for
end for

$\mathcal{Z} \leftarrow \mathcal{Z}^{\mathtt{L}}$                                  ▷ Formulation of $\mathcal{Z}$
for $n$ in $\mathcal{N}$ do
    $\mathcal{Z}' \leftarrow \{\}$
    for $\mathcal{Z}_i$ in product($\mathcal{Z}, \mathcal{Z}^{\mathtt{V}}_n$) do
        if HASNONEMPTYPARTITION($\mathcal{Z}_i$) then
            $\mathcal{Z}'$.ADD($\mathcal{Z}_i$)
        end if
    end for
    $\mathcal{Z} \leftarrow \mathcal{Z}'$
end for
return $\mathcal{Z}$

function HASNONEMPTYPARTITION($\mathcal{Z}_i$)
    for $k = 0$ to $k_h$ do
        if part($k; \mathcal{Z}_i$) $\neq \emptyset$ then
            return True
        end if
    end for
    return False
end function
```

D. Computation of Reachable Sets

In addition to our previous works [1], [12], we semantically label the reachable sets and constrain them to states satisfying the desired specifications (Alg. 2 lines 4–6). As the reachable sets are computed iteratively over time, it suffices to give a detailed explanation for one step of the computation.

1) Propagation of Base Sets (Alg. 2, line 3): Each base set $\mathcal{R}^{(i)}_{k-1} \in \mathcal{R}_{k-1}$ of the previous time step is propagated according to the system model (1), resulting in the propagated sets $\mathcal{R}^{\mathtt{P},(i)}_k \in \mathcal{R}^{\mathtt{P}}_k$ (see Fig. 5a). The propagation is performed similarly to the method described in [1], except that one can additionally impose acceleration constraints. As an example, rule R_G2 [15] describes situations in which braking abruptly is allowed, i.e., braking harder than a predefined value $a_{\mathtt{def}} >$

Algorithm 2 Computation of Reachable Sets

Inputs: Specification ϕ, base sets $\mathcal{R}_0^{(i)} \in \mathcal{R}_0$ with their semantic labels $\mathcal{L}_0^{(i)} \in \mathcal{L}_0$, planning horizon k_h, realizable sets of position predicates $\mathcal{Z}$.

Output: Reachability graph $\mathcal{G}_\mathcal{R}$.

```
 1: $\mathcal{G}_\mathcal{R}$.ADD($\mathcal{R}_0, \mathcal{L}_0$)
 2: for $k = 1$ to $k_h$ do
 3:     $\mathcal{R}_k^{\mathtt{P}} \leftarrow$ PROPAGATE($\mathcal{R}_{k-1}$)                      ▷ Sec. III-D.1
 4:     $\mathcal{R}_k^{\mathtt{S}} \leftarrow$ SPLITTING($\mathcal{R}_k^{\mathtt{P}}, \mathcal{Z}$)                    ▷ Sec. III-D.2
 5:     $\mathcal{R}_k^{\mathtt{L}}, \mathcal{L}_k \leftarrow$ LABELING($\mathcal{R}_k^{\mathtt{S}}, \phi$)                ▷ Sec. III-D.3
 6:     $\mathcal{R}_k^{\mathtt{C}} \leftarrow$ CHECKCOMPLIANCE($\mathcal{R}_k^{\mathtt{L}}, \mathcal{L}_k, \phi$)        ▷ Sec. III-D.4
 7:     $\mathcal{R}_k \leftarrow$ CREATENEWBASESETS($\mathcal{R}_k^{\mathtt{C}}$)             ▷ Sec. III-D.5
 8:     $\mathcal{G}_\mathcal{R}$.ADD($\mathcal{R}_k, \mathcal{L}_k$)
 9: end for
10: return $\mathcal{G}_\mathcal{R}$
```

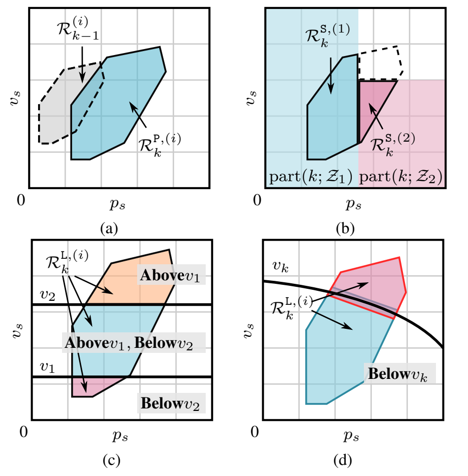

$\mathcal{R}_{k-1}^{(i)}$
$\mathcal{R}_k^{\mathtt{P},(i)}$
v_s
p_s
0
(a)
$\mathcal{R}_k^{\mathtt{S},(1)}$
$\mathcal{R}_k^{\mathtt{S},(2)}$
part($k; \mathcal{Z}_1$)
part($k; \mathcal{Z}_2$)
(b)
$\mathcal{R}_k^{\mathtt{L},(i)}$
v_2
v_1
Abovev_1
Abovev_1, **Below**v_2
Belowv_2
(c)
v_k
$\mathcal{R}_k^{\mathtt{L},(i)}$
Belowv_k
(d)

Fig. 5: Propagation, splitting, and labeling of base sets. For clarity, we only show the operations in the s-direction. Labels of polytopes are shown in gray boxes. (a) Propagation. (b) Splitting regarding partitions. (c) Splitting regarding static velocity predicates: three polytopes split with predicates **Above**v_1 and **Below**v_2. (d) Splitting regarding dynamic velocity predicates: two polytopes split with a predicate **Below**v_k. Notably, the two newly split polytopes are slightly over-approximated and convexified due to the nonlinearity introduced by the predicate.

$\underline{a}_s$. If this rule is considered, we only propagate the base sets with the acceleration interval $[\underline{a}_s, \overline{a}_s]$ under the specified situations, otherwise $[a_{\mathtt{def}}, \overline{a}_s]$. Sets $\mathcal{L}_k^{(i)}$ of propagated sets $\mathcal{R}_k^{\mathtt{P},(i)}$ are initialized with empty sets.

2) Splitting (Alg. 2, line 4): The propagated sets $\mathcal{R}_k^{\mathtt{P},(i)}$ are split into new sets $\mathcal{R}_k^{\mathtt{S},(i)} \in \mathcal{R}_k^{\mathtt{S}}$ regarding position and velocity predicates. We first introduce the velocity predicates.

a) Static Velocity Predicates: These predicates typically relate to the constant extremum requirements on velocities. Rule R_G3 [15] requires that maximum velocity limits originating from various sources to be respected. These include limits introduced due to the limited field of view of the ego vehicle and vehicle type-specific limits.

b) Dynamic Velocity Predicates: These predicates depend on other dynamic obstacles present in the scenario. Examples are predicates indicating whether the ego vehicle is driving at a safe velocity regarding a leading or a following vehicle V_n [15, cf. Sec. IV.C].

The splitting of $\mathcal{R}_k^{\mathtt{P},(i)}$ is performed as follows:

1) $\mathcal{R}_k^{\mathtt{P},(i)}$ are split such that the newly split sets intersect only with a single partition of $\mathcal{Z}$ (see Fig. 5b).
2) The split sets are further split, over-approximated and convexified (if needed) regarding velocity predicates (see Fig. 5c–d).

3) Semantic Labeling (Alg. 2, line 5): Next, we evaluate the general traffic situation predicates introduced by the specification (see details in Sec. III-D.4) on $\mathcal{R}_k^{\mathtt{S},(i)}$, and update their semantic labels $\mathcal{L}_k^{(i)}$. These predicates may reveal whether the ego vehicle has conducted a lanelet-change maneuver (see Sec. IV-A) and if it is stuck in a traffic congestion, etc. The labels $\mathcal{L}_k^{(i)}$ are updated as follows:

1) Sets $\mathcal{R}_k^{\mathtt{S},(i)}$ propagated with acceleration-specific rules include corresponding predicates in their sets of semantic labels $\mathcal{L}_k^{(i)}$.
2) Sets $\mathcal{R}_k^{\mathtt{S},(i)}$ include the position predicates associated with the partition with which it intersects, velocity predicates, and general traffic situation predicates that evaluate to `True` in their sets of semantic labels $\mathcal{L}_k^{(i)}$.

The sets with updated $\mathcal{L}_k^{(i)}$ are denoted by $\mathcal{R}_k^{\mathtt{L},(i)} \in \mathcal{R}_k^{\mathtt{L}}$.

4) Checking Specification Compliance (Alg. 2, line 6): In this step, we iterate through $\mathcal{R}_k^{\mathtt{L},(i)}$ and examine the compliance of their labels $\mathcal{L}_k^{(i)}$ with the given specification. Let σ be an atomic proposition, a time-labeled propositional formula ϕ is defined in Backus-Naur form as:

$$\phi ::= \sigma \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid \mathrm{G}_I(\phi),$$

where the operator G_I dictates a time interval I for which ϕ should hold. If I is unspecified, we assume it to be the entire planning horizon $[0, k_h]$. Let $\mathcal{L}_k^{(i)}$ be the set of labels to be examined, its compliance with σ is defined as: $\mathcal{L}_k^{(i)} \models \sigma$ iff $\sigma \in \mathcal{L}_k^{(i)}$. As an example, the following specification enforces the ego vehicle to follow V_1 between time steps 0 and 10, and never to be on its right for the entire planning horizon:

$$\mathrm{G}_{[0,10]}(\mathbf{Behind}(V_1) \wedge \mathbf{AlignedWith}(V_1))\wedge$$
$$\mathrm{G}(\neg\mathbf{RightOf}(V_1)).$$

We discard $\mathcal{R}_k^{\mathtt{L},(i)}$ whose set of semantic labels $\mathcal{L}_k^{(i)}$ do not comply with ϕ, and refer to the remaining sets as $\mathcal{R}_k^{\mathtt{C},(i)} \in \mathcal{R}_k^{\mathtt{C}}$. Recall that the reachable sets enclose all drivable trajectories of the vehicle, thus an empty set $\mathcal{R}_k^{\mathtt{C}}$ implies that ϕ is unsatisfiable and cannot be complied with by any possible trajectory of the ego vehicle. In such a case, one can either recompute the reachable sets with a different specification, or trigger previously computed fail-safe trajectories [21]. Integrating MTL specifications into our reachable sets using model checkers will be a future study.

5) Creation of New Base Sets (Alg. 2, line 7): Finally, the new base sets $\mathcal{R}_k^{(i)} \in \mathcal{R}_k$ are created from the nonempty sets $\mathcal{R}_k^{\mathtt{C},(i)}$. The substeps include obtaining the drivable areas $\mathcal{D}_k^{\mathtt{C},(i)}$ of $\mathcal{R}_k^{\mathtt{C},(i)}$, merging and repartitioning $\mathcal{D}_k^{\mathtt{C},(i)}$, and ultimately producing the sets $\mathcal{R}_k^{(i)}$. These are performed similarly to the description in [1, Alg. 1] with one difference: to preserve the set of position predicates at the merging step, only $\mathcal{D}_k^{\mathtt{C},(i)}$ projected from sets $\mathcal{R}_k^{\mathtt{C},(i)}$ with the same partition are merged. The reachability graph $\mathcal{G}_{\mathcal{R}}$ is updated in the end by inserting sets $\mathcal{R}_k^{(i)}$ along with their labels $\mathcal{L}_k^{(i)}$ as new nodes.

IV. Evaluation

In this section, we exhibit the applicability of our method using varied specifications on three scenarios. Selected parameters and computation results are listed in Tab. II. The animation of the evaluation can be found at https://mediatum.ub.tum.de/1595757. Before presenting the evaluation results, we introduce relevant labels and explain how selected traffic rule elements can be incorporated.

A. Relevant Labels

- **AdmissibleBraking**: This indicates that the rule R_G2 is considered (see Sec. III-D.1).
- **SafeFollowingVelocity**, **SafeLeadingVelocity**: These indicate that the ego vehicle has respected safe following/leading velocities to other dynamic obstacles, respectively (see Sec. III-D.2).
- **ChangeLanelet**(L_1, L_2): This indicates that the ego vehicle has performed a lanelet-change maneuver from L_i to L_j.

B. Incorporating Traffic Rule Elements

1) Prohibiting Change of Lanelet: Assuming a case where the ego vehicle is not allowed to change from L_1 to L_2, which may be imposed by different traffic rule elements, such as road markings, no-overtaking signs, and traffic lights. We model this with $\mathrm{G}_I(\neg\textbf{ChangeLanelet}(L_1, L_2))$.

2) Lanelet-specific Velocity Limits: Lanelet-specific velocity limits can neither be modeled as static nor dynamic predicates (as described in Sec. III-D.2). Recalling that the partitions of sets of position predicates are modeled with hyperrectangles $\mathtt{rect}_q$, we adjust the velocity intervals $[\underline{v}_{q,s}, \overline{v}_{q,s}]$ of $\mathtt{rect}_q$ in the lanelets to incorporate these velocity limits.

C. Scenario I: Precise Overtaking

The first scenario illustrates a situation where the ego vehicle should overtake a leading vehicle V_1 in the presence of another vehicle V_2. The following specification is issued, for example, by a high-level maneuver planner, which should be precisely followed by the ego vehicle:

$$\mathrm{G}_{[0,15]}(\textbf{Behind}(V_1) \wedge \textbf{AlignedWith}(V_1)) \wedge$$
$$\mathrm{G}_{[16,38]}(\textbf{InLanelet}(L_2) \vee \textbf{InLanelet}(L_4)) \wedge$$
$$\mathrm{G}_{[39,45]}(\textbf{InFrontOf}(V_1) \wedge \textbf{Behind}(V_2) \wedge \textbf{InLanelet}(L_3)) \wedge$$
$$\mathrm{G}(\textbf{AdmissibleBraking}).$$

TABLE II: Selected Parameters and Computation Results

Description	Unit	I		II		III	
Parameter							
k_h	step	45	45	40	40	40	40
Δt	s	0.1	0.1	0.1	0.1	0.1	0.1
$\overline{v}_s$	m/s	16.6	16.6	16.6	16.6	16.6	16.6
$\underline{v}_s$	m/s	0	0	0	0	0	0
$v_{s,0}$	m/s	12.0	12.0	14.5	14.5	14.0	14.0
$\overline{v}_d$	m/s	4.0	4.0	4.0	4.0	4.0	4.0
$\underline{v}_d$	m/s	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
$\overline{a}_s$	m/s^2	2.0	2.0	2.0	2.0	2.0	2.0
$\underline{a}_s$	m/s^2	-6.0	-6.0	-6.0	-6.0	-6.0	-6.0
$a_{\mathtt{def}}$	m/s^2	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0
Grid cell size	m^2	1×1	2×2	1×1	2×2	1×1	2×2
State Space Partition							
$\lvert\mathcal{Z}\rvert$	-	14	14	14	14	52	52
# Hyperrectangles	-	1800	450	1800	450	1700	430
Computation Time							
Propagation	ms	360	131	434	213	288	174
Splitting	ms	439	138	715	296	1360	532
Labeling	ms	25	9	95	20	85	37
Compliance check	ms	92	31	123	56	268	112
Creation of new base sets	ms	16	8	30	17	43	22
Sum	ms	932	317	1397	602	2044	877

We compute the reachable sets over time as explained in the previous section. The resulting reachable sets are nonempty, thereby implying that one may find a trajectory that satisfies thet specification. Fig. 6 visualizes the drivable areas of the ego vehicle at different time steps, as well as an exemplary trajectory planned within the reachable sets [2]. Changing **Behind**(V_2) to **InFrontOf**(V_2) for $k \in [39, 45]$ in the specification yields an empty reachable set; thus, we can reject it before trying to plan a trajectory that satisfies the specification.

D. Scenario II: Respecting Safe Velocities

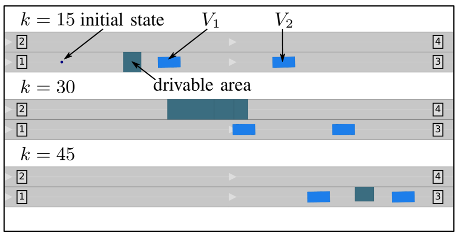
In this scenario, the ego vehicle is driving on the left side of two other vehicles V_1 and V_2, and wishes to change to lanelets on the right (1 and 3). The unrestricted reachable sets (see Fig. 7a) show that the ego vehicle can reach lanelet 1 at time step $k = 20$ and deliberately cut in between V_1 and V_2 for $k \in [30, 40]$. By contrast, after considering the specification with safe velocity rules

$$\mathrm{G}(\textbf{SafeFollowingVelocity} \wedge$$
$$\textbf{ChangeLanelet}(\cdot,\cdot) \rightarrow \textbf{SafeLeadingVelocity}),$$

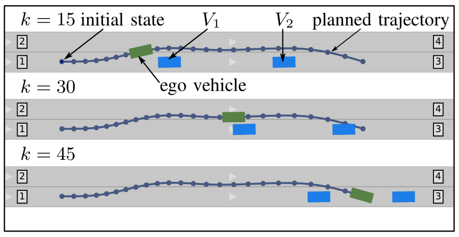
being in lanelet 1 is no longer legal at time step $k = 20$, so is the case with being between V_1 and V_2 for $k \in [30, 40]$ (see Fig. 7b). Following V_1 in lanelets 1 and 3 is still a legal maneuver at later time steps, provided the ego vehicle has sufficiently slowed down to respect the safe following velocity rule.

E. Scenario III: Overtaking from the Left

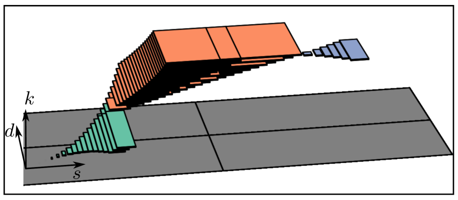
Next, we present a scenario in which the ego vehicle intends to overtake its preceding vehicle V_1. The unrestricted reachable sets propagate to all lanelets in the scenario and enclose various maneuvers, including overtaking V_1 from the right (see Fig. 8a). By imposing the constraint that the ego



(a)

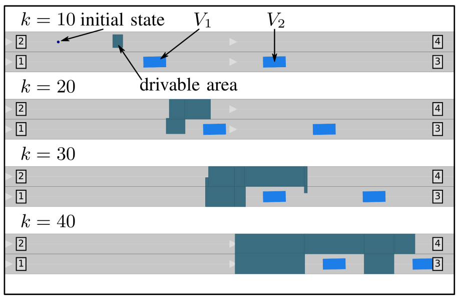


(b)

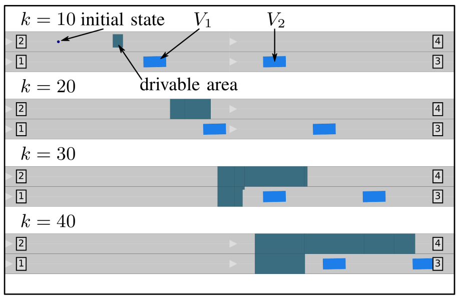


(c)

Fig. 6: Scenario I: Precise overtaking. (a) Drivable area at different time steps. (b) Exemplary trajectory planned within the reachable set. (c) Drivable area over time. Each color corresponds to a clause in the specification whose time interval is specified.



(a)



(b)

Fig. 7: Scenario II: Respecting safe velocities to other vehicles. (a) Not considering specifications. (b) Considering specifications.

vehicle should overtake V_1 at some future time steps, and not being on its right for the entire planning horizon, we explicitly demand that the ego vehicle can only overtake V_1 from its left. In addition, we add a constraint to forbid entering lanelet 4 to further restrict the reachable sets:

$$\begin{aligned}&\mathrm{G}_{[35,45]}(\textbf{InFrontOf}(V_1) \land \textbf{AlignedWith}(V_1)) \land \\ &\mathrm{G}(\neg\textbf{RightOf}(V_1) \land \neg\textbf{ChangeLanelet}(L_3, L_4)).\end{aligned}$$

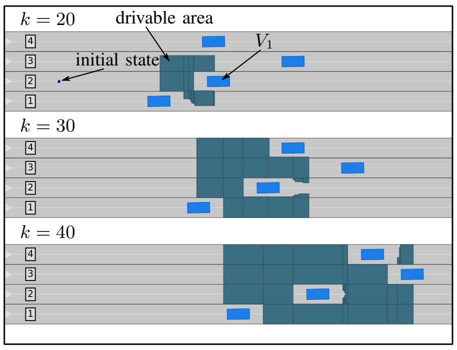
As can be seen from Fig. 8b, the reachable sets flow to lanelet 3 from the left of V_1 at $k = 20$, return back to lanelet 2 at $k = 30$, and finally end in front of V_1 at $k = 40$, which is exactly required by the specification.

F. Analysis of Computation Results

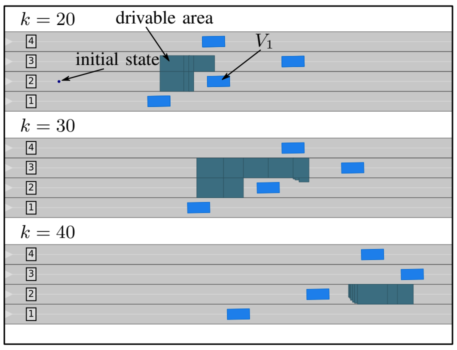
The computation times listed in Tab. II are obtained through a prototype implemented using Python and C++ on a 2.8 GHz laptop. The computation times for operations, including splitting, labeling, and compliance checking, are linear to the number of nodes in the reachability graph, which is, in turn, proportional to the number of hyperrectangles: For all three scenarios, the evaluations were performed using two grid cell sizes. By doubling the cell sizes and thereby reducing the number of hyperrectangles, we observed a drastic decrease (approximately 60%) in the computation time, which could be further improved by representing the partitions with more sophisticated polytopes. We refer to [1, Appendix B] for a detailed explanation of the computation complexity of propagation and creation of base sets. The method for accelerating the reachable set computation in [22] did not consider dynamic velocity and acceleration constraints; how this method can be coupled with our reachability analysis should be investigated in the future.

V. Conclusions

In this study, we proposed a method to obtain specification-compliant reachable sets for a considered ego vehicle, which is used to guide motion planners to find specification-compliant trajectories. Compared with existing methods, the proposed method can not only consider traffic and handcrafted rules considering position predicates but also velocity, acceleration, and general traffic situation predicates. The evaluations showed that our method could easily incorporate desired specifications as well as identify and reject the unsatisfiable ones. In the future, we will investigate how to fully incorporate traffic rules formulated in MTL formulas into our reachable sets. MTL formulas are much more expressive than propositional logic and LTL by having temporal operators over time, such as $\textbf{Once}_I$ and



(a)



(b)

Fig. 8: Scenario III: Overtaking from the left. (a) Not considering specifications. (b) Considering specifications.

Future$_I$. In addition, it is also worth investigating that how specifications should be manipulated when they cannot be fully complied with, e.g., when a collision cannot be avoided without making a prohibited lane change.

Acknowledgments

Special thanks to Stefanie Manzinger for her support to this work. This work was funded by the Deutsche Forschungsgemeinschaft (German Research Foundation) within the Priority Program SPP 1835 *Cooperative Interacting Automobiles* under grant No. AL 1185/4-2, and within the Huawei-TUM collaboration project *Research on Key Technologies of Safety Assurance for Autonomous Vehicles*. The authors appreciate the fruitful collaboration with the project partners.

References

[1] S. Söntges and M. Althoff, "Computing the drivable area of autonomous road vehicles in dynamic road scenes," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 6, pp. 1855–1866, 2018.

[2] S. Manzinger, C. Pek, and M. Althoff, "Using reachable sets for trajectory planning of automated vehicles," *IEEE Trans. Intell. Veh.*, 2020 (to appear).

[3] Y. Shoukry, P. Nuzzo, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada, "SMC: satisfiability modulo convex programming," *Proc. of the IEEE*, vol. 106, no. 9, pp. 1655–1679, 2018.

[4] M. Lahijanian, M. R. Maly, D. Fried, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi, "Iterative temporal planning in uncertain environments with partial satisfaction guarantees," *IEEE Trans. Robot.*, vol. 32, no. 3, pp. 583–599, 2016.

[5] A. Bhatia, M. R. Maly, L. E. Kavraki, and M. Y. Vardi, "Motion planning with complex goals," *IEEE Robot. Autom. Mag.*, vol. 18, no. 3, pp. 55–64, 2011.

[6] Y. Zhou, D. Maity, and J. S. Baras, "Timed automata approach for motion planning using metric interval temporal logic," in *Proc. of Eur. Control Conf.*, 2016, pp. 690–695.

[7] E. Plaku and S. Karaman, "Motion planning with temporal-logic specifications: progress and challenges," *AI Commun.*, vol. 29, no. 1, pp. 151–162, 2015.

[8] M. Hekmatnejad, S. Yaghoubi, A. Dokhanchi, H. B. Amor, A. Shrivastava, L. Karam, and G. Fainekos, "Encoding and monitoring responsibility sensitive safety rules for automated vehicles in signal temporal logic," in *Proc. of the ACM/IEEE Int. Conf. Formal Method. Model. Syst. Des.*, 2019, pp. 1–11.

[9] A. Rizaldi, F. Immler, and M. Althoff, "A formally verified checker of the safe distance traffic rules for autonomous vehicles," in *NASA Formal Method. Symp.*, 2016, pp. 175–190.

[10] K. Esterle, V. Aravantinos, and A. Knoll, "From specifications to behavior: maneuver verification in a semantic state space," in *Proc. of the IEEE Intell. Veh. Symp.*, 2019, pp. 2140–2147.

[11] A. Best, S. Narang, D. Barber, and D. Manocha, "Autonovi: autonomous vehicle planning with dynamic maneuvers and traffic constraints," in *Proc. of the IEEE Int. Conf. Intell. Robot. Syst.*, 2017, pp. 2629–2636.

[12] S. Söntges and M. Althoff, "Computing possible driving corridors for automated vehicles," in *Proc. of the IEEE Intell. Veh. Symp.*, 2017, pp. 160–166.

[13] R. Kohlhaas, T. Bittner, T. Schamm, and J. M. Zöllner, "Semantic state space for high-level maneuver planning in structured traffic scenes," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2014, pp. 1060–1065.

[14] K. Esterle, L. Gressenbuch, and A. Knoll, "Formalizing traffic rules for machine interpretability," in *Proc. of the IEEE Connect. Autom. Veh. Symp.*, 2020, pp. 1–7.

[15] S. Maierhofer, A.-K. Rettinger, E. C. Mayer, and M. Althoff, "Formalization of interstate traffic rules in temporal logic," in *Proc. of the IEEE Intell. Veh. Symp.*, 2020, pp. 752–759.

[16] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: composable benchmarks for motion planning on roads," in *Proc. of the IEEE Intell. Veh. Symp.*, 2017, pp. 719–726.

[17] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *Proc. of the IEEE Intell. Veh. Symp.*, 2014, pp. 420–425.

[18] M. Koschi and M. Althoff, "Set-based prediction of traffic participants considering occlusions and traffic rules," *IEEE Trans. Intell. Veh.*, 2020 (to appear).

[19] C. Pek, V. Rusinov, S. Manzinger, M. C. Üste, and M. Althoff, "CommonRoad Drivability Checker: Simplifying the development and validation of motion planning algorithms," in *Proc. of the IEEE Intell. Veh. Symp.*, 2020, pp. 1013–1020.

[20] F. Altché and A. De La Fortelle, "Partitioning of the free space-time for on-road navigation of autonomous ground vehicles," in *Proc. of the IEEE Conf. Decis. Control*, 2017, pp. 2126–2133.

[21] C. Pek and M. Althoff, "Fail-safe motion planning for online verification of autonomous vehicles using convex optimization," *IEEE Trans. Robot.*, 2020 (to appear).

[22] M. Klischat and M. Althoff, "A multi-step approach to accelerate the computation of reachable sets for road vehicles," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2020, pp. 1–7.

A.2 Specification-Compliant Driving Corridors for Motion Planning of Automated Vehicles [2]

Summary For automated vehicles to effectively and safely participate in road traffic, strict compliance with legal specifications including traffic rules is essential. This work addresses the motion planning problem for automated vehicles considering specifications expressed in metric temporal logic.

To this end, we couple set-based reachability analysis with automata-based model checking to generate specification-compliant driving corridors. Reachability analysis determines the set of states reachable by a vehicle over time (referred to as *reachable sets*). Computing reachable sets in an over-approximative fashion allows one to identify all collision-free driving corridors of a vehicle. A driving corridor is a timed sequence of position and velocity bounds that can be utilized by motion planners to significantly reduce the planning space. Given a set of desired behavioral specifications, the model checking technique verifies the specifications on a suitable model of a given system through systematic inspection of all states of the model. Through coupling reachability analysis with model checking, we efficiently identify all driving corridors of a vehicle that are both collision-free and compliant with considered specifications. The constraints extracted from such driving corridors can be applied to motion planners to expedite the generation of specification-compliant trajectories.

This article provides several contributions: (1) Building on the work of [1], we incorporate temporal logic specifications (including interstate and intersection traffic rules) into the reachability analysis of automated vehicles. (2) We couple reachability analysis and model checking to identify collision-free and specification-compliant driving corridors. (3) We generate a product graph from which the optimal driving corridor can be separately determined using arbitrary utility functions.

Our approach exhibits the following properties: (1) Unlike conventional motion planners, our reachable set computation requires less time in more critical scenarios with smaller solution space. (2) We perform efficient and exhaustive verification of all driving corridors of an automated vehicle against considered specifications, thanks to mature model checking techniques. (3) Our approach detects conflicting or non-satisfiable specifications prior to the motion planning phase. (4) It is applicable to traffic scenarios involving static and dynamic obstacles of arbitrary shapes. (5) The total computation time consumes only a fraction of the planning horizon.

The experiments demonstrate that our approach can be easily integrated into motion planners to efficiently derive trajectories that comply with temporal specifications, particularly in scenarios with a narrow solution space. Moreover, our computation of reachable sets only requires a fraction of the planning horizon, as demonstrated by benchmarking against over 50 CommonRoad scenarios, indicating the real-time capability of our approach.

Contributions of E. I. L. E. I. L. developed the idea of the research (together with M. A.); E. I. L. designed, conducted, and evaluated the experiments; E. I. L. wrote the article (together with M. A.).

Journal article The accepted version of the article is reprinted. The final version of record is available at `https://doi.org/10.1109/TIV.2023.3289580`.

Specification-Compliant Driving Corridors for Motion Planning of Automated Vehicles

Edmond Irani Liu and Matthias Althoff

***Abstract*—It is crucial for automated vehicles to explicitly comply with specifications, including traffic rules, to ensure their safe and effective participation in road traffic. Such compliance is also essential for vehicle manufacturers to avoid liability claims in the event of accidents. We propose a novel approach addressing the problem of specification-compliant motion planning for automated vehicles. Our approach couples set-based reachability analysis with automata-based model checking and outputs specification-compliant driving corridors. These driving corridors serve as motion planning constraints and expedite the generation of trajectories complying with specifications expressed in metric temporal logic. In contrast to existing works, our approach efficiently and exhaustively verifies all driving corridors of an automated vehicle, leveraging mature model checking techniques. We demonstrate the applicability, effectiveness, and efficiency of our approach using various specifications on scenarios from the CommonRoad benchmark suite. Moreover, we benchmark the performance of our prototype against multiple scenarios, indicating that our approach is real-time capable.**

***Index Terms*—automated vehicles, motion planning, traffic rules, temporal logic, reachability analysis, model checking.**

I. Introduction

Automated vehicles are expected to explicitly comply with traffic rules to safely and effectively participate in mixed road traffic, where both automated and human-driven vehicles coexist. In addition, automated vehicle manufacturers bear the responsibility to certify such compliance and by this avoid liability claims in the event of accidents. Despite the importance of this matter, most previous studies on motion planning of automated vehicles reported in recent surveys [1]–[4] either entirely disregard traffic rules or only consider a limited fraction of them. This is due to the sheer difficulty of formalizing traffic rules in a machine-interpretable way and their integration into motion planners. In this article, the term *specifications* refers to traffic rules and other requirements formalized in temporal logic to which vehicles must adhere. Examples of such formalizations can be found in [5]–[9].

Generating drivable trajectories for vehicles complying with specifications involves reasoning with both their continuous and discrete states. The former typically contains the position, velocity, and orientation of a vehicle; examples of the latter are the operation mode of the vehicle and its logical relation to other traffic participants. Computational challenges arise in generating such trajectories due to factors such as vehicle dynamics, considered specifications (including collision avoidance), and the interdependence of planned trajectories and constraints originating from the specifications [10]. Per recent surveys [1]–[4], no approach exists that plans specification-compliant motions in continuous state space: Classical motion planners generate collision-free and dynamically feasible trajectories but cannot guarantee specification compliance; Also, planning in a discretized state space may output discrete plans that satisfy the specifications but disregard drivability constraints or lead to collisions.

Both authors are with the School of Computation, Information and Technology, Technical University of Munich, 80333 Munich, Germany (e-mail: edmond.irani@tum.de, althoff@tum.de).

We propose a novel and efficient approach addressing the problem of specification-compliant motion planning for automated vehicles using set-based reachability analysis and automata-based model checking. *Reachability analysis* is a technique for determining the set of states reachable by a system over time (henceforth referred to as *reachable set*), starting from a set of initial states. Computing the reachable sets of a vehicle in an over-approximative fashion enables the exploration of its continuous state space and the identification of all its collision-free *driving corridors* [11]–[13]. A driving corridor represents a timed sequence of position and velocity bounds that can be utilized by motion planners to significantly reduce the planning space, especially in situations with a narrow solution space [12]–[14]. *Model checking* is a formal verification technique that verifies desired behavioral specifications on a suitable model of a given system through systematic inspection of all states of the model. By coupling reachability analysis with model checking, we efficiently identify all driving corridors of automated vehicles that are both collision-free and compliant with enforced specifications. Applying constraints extracted from such driving corridors to motion planners expedites the generation of trajectories complying with enforced specifications.

A. Related Work

We categorize existing works on specification-compliant motion planning based on *when* specifications are considered:

1) Considering Compliance After Motion Planning: Runtime verification, also known as *monitoring*, refers to checking whether an execution of a system meets the expected behaviors. For instance, a monitor for examining the compliance of vehicles with safe distance rules and overtaking rules is presented in [9]. While the monitoring is often efficient, monitors typically only return a *robustness degree* (the extent of satisfaction of specifications) or a verdict (*true* or *false*) on whether the specifications have been satisfied. No alternative trajectory is returned if a trajectory is deemed inferior or rejected by a monitor. This generally leads to replanning and verifying many trajectories for more complex specifications before finding a specification-compliant solution.

Instead of examining individual trajectories, it is also possible to verify infinitely many trajectories at once: The work in [15] describes a method for model checking reachable sets of continuous and hybrid systems against *signal temporal logic* [16] specifications. As with monitoring, it only returns a verdict (possibly with counterexamples in case of violation), which has limited usage for our motion planning application.

2) Considering Compliance During Motion Planning: Existing efforts in this category can be roughly divided into three groups: multilayered approaches, approaches based on *mixed-integer linear programming* (MILP), and approaches based on *rapidly exploring random trees* (RRTs) [17]. Multilayered approaches [18]–[27] commonly handle specification-compliant motion planning problems using a high-level discrete planning layer and a low-level trajectory planning layer. The discrete planning layer relies on discrete abstractions of the system of interest and generates plans satisfying the specifications, which guide the trajectory planning process at a later stage. The discrete plans are generated based on, among others, automata theory [20], [22]–[25], [27], satisfiability modulo theory [18], [21], and monitors [19]. For instance, article [24] adopts timed automata to generate timed paths that satisfy *metric temporal logic* (MTL) [28] specifications for indoor robot navigation; the work in [21] introduces a satisfiability modulo convex programming framework that handles both convex constraints over continuous states and Boolean constraints over discrete states for cyber-physical systems; in [19], the authors obtain high-level driving maneuvers of automated vehicles that respect traffic rules in *linear temporal logic* (LTL) [29] via monitoring. In most cases, the dynamic constraints of the system are not considered in the discrete plans; thus, the drivability of the plans is often not ensured. Consequently, frequent replanning in both the discrete and trajectory planning layers can be expected, especially in complex and highly dynamic environments.

The basic idea of MILP-based approaches is to cast temporal logic specifications as mixed-integer linear constraints. After introducing system dynamic constraints, a solver generates a specification-compliant trajectory while optimizing certain cost functions. MILP problems are NP-hard in nature [30, Ch. 11], and the constraints mentioned above bring about auxiliary decision variables that exponentially increase the complexity and solution time of the optimization problem (e.g., see [31]–[35]). This is often a limiting factor for applications with high real-time requirements such as motion planning of automated vehicles.

RRT-based approaches typically generate specification-compliant trajectories in an incremental manner. The works in [22], [36]–[40] build on the RRT* algorithm [41], which is an asymptotically optimal variant of the well-known RRT algorithm. The growth of the tree is steered or pruned, e.g., using automata [22], [36], [38], [40] or robustness degrees [37], [39] of the specifications. Given enough time and iterations, a trajectory respecting the system dynamics and specifications can be found. While RRT-based methods provide fast solutions to specific problems, they are not well-suited for safety-critical applications due to their inherent characteristic known as *probabilistic completeness* [17], [41]. Moreover, the performance of RRT-based methods typically degrades in situations with a narrow solution space [42].

B. Contributions

Our approach provides the following contributions:

- Extension of [43] by integrating temporal logic specifications (including interstate and intersection traffic rules) into the reachability analysis of automated vehicles.
- Coupling reachability analysis with model checking for identifying collision-free and specification-compliant driving corridors. Such corridors expedite the generation of specification-compliant trajectories for motion planners that accept position and velocity constraints.
- Generation of a *product graph* from which the optimal driving corridor can be determined using arbitrary utility functions in a separate stage.

Our approach has the following properties:

- In contrast to conventional motion planners, our reachable set computation requires less time in more critical scenarios: The computation can be performed the faster, the smaller the solution space is, which is often the case in critical scenarios.
- Efficient and exhaustive verification of all driving corridors of an automated vehicle against considered specifications, owing to mature model checking techniques.
- Detection of conflicting or non-satisfiable specifications before motion planning.
- Applicability in traffic scenarios involving static and dynamic obstacles of arbitrary shapes.
- The total computation time requires only a fraction of the planning horizon.

The remainder of this article is organized as follows: After presenting the preliminaries and problem statement in Sec. II, we describe our methodology in Sec. III. The implementation of our reachability analysis is detailed in Sec. IV, followed by the evaluation of predicates and the rewriting of specifications in Sec. V and Sec. VI, respectively. In Sec. VII, we elaborate on the identification of specification-compliant driving corridors. Our approach is evaluated in Sec. VIII and we finish with conclusions in Sec. IX.

II. Preliminaries and Problem Statement

After introducing the necessary preliminaries, including the general setup, temporal logics to formalize our specifications, set-based reachability analysis, automata-based model checking, and driving corridors, we present the problem statement.

A. General Setup

The vehicle for which trajectories should be planned is referred to as the *ego vehicle*. The road network consists of lanelets [44], each modeled with polylines representing its left and right boundaries. We assume a high-level route planner is available that plans a route through the road network, whose centerline is considered as the reference path $\Gamma : \mathbb{R} \rightarrow \mathbb{R}^2$. A local curvilinear coordinate system $F^{\mathtt{L}}$ of the ego vehicle is constructed from the reference path as described in [44], within which (s, d) describes the longitudinal coordinate s along the reference path and the lateral coordinate d orthogonal to $\Gamma(s)$. The adoption of $F^{\mathtt{L}}$ facilitates the formulation of maneuvers from the perspective of the ego vehicle, such as following a lane and stopping before a stop line. We denote by $k \in \mathbb{N}_0$ a step corresponding to time $t_k = k\Delta_t$, with $\Delta_t \in \mathbb{R}_+$ being a predefined time increment. Motions of the ego vehicle are planned up to the planning horizon $k_h \in \mathbb{N}$, whose dynamics is

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k), \tag{1}$$

where $\boldsymbol{x}_k \in \mathcal{X}_k \subset \mathbb{R}^{n_x}$ represents the state of the ego vehicle in the state space $\mathcal{X}_k$, $\boldsymbol{u}_k \in \mathcal{U}_k \subset \mathbb{R}^{n_u}$ represents an input in the input space $\mathcal{U}_k$. A possible input trajectory over time is denoted by $\boldsymbol{U}$. We also denote by τ_k the valuation of the ego vehicle with state $\boldsymbol{x}_k$ over atomic propositions $\mathcal{AP}$ (see Sec. II-B), each of which indicates a logical relation between the ego vehicle and entities in an environment model such as lanes and obstacles (later detailed in Sec. V).

B. Temporal Logics

Specifications considered in this work are expressed in *MTL with past over finite traces* (MTLp$_f$) [28], [45]. MTLp$_f$ shares the same syntax with MTL and is interpreted over traces of *finite* length. We settle on MTLp$_f$ since (a) it is expressive enough to formulate traffic rules with timing constraints, e.g., see [5]–[7], and (b) traces in our system have finite length.

1) Metric Temporal Logic with Past over Finite Traces: An MTLp$_f$ formula $\varphi^{\mathtt{M}}$ over atomic propositions $\mathcal{AP}$ has the following syntax given in Backus-Naur form [28], [45]:

$$\varphi^{\mathtt{M}} ::= \sigma \,|\, \neg\varphi^{\mathtt{M}} \,|\, \varphi_1^{\mathtt{M}} \wedge \varphi_2^{\mathtt{M}}, \,|\, \mathbf{X}_I\varphi^{\mathtt{M}} \,|\, \varphi_1^{\mathtt{M}}\mathbf{U}_I\varphi_2^{\mathtt{M}} \,|\, \mathbf{Y}_I\varphi^{\mathtt{M}} \,|\, \varphi_1^{\mathtt{M}}\mathbf{S}_I\varphi_2^{\mathtt{M}},$$

where $\sigma \in \mathcal{AP}$ is an atomic proposition, $\neg$ (Not) and $\wedge$ (And) are Boolean connectives, **X** (ne**X**t) and **U** (**U**ntil) are future-time connectives, **Y** (**Y**esterday) and **S** (**S**ince) are past-time connectives, and $I = [a, b]$ is a bounded interval. Without loss of generality, we assume $a, b \in \mathbb{N}_0$. We also use the following common abbreviations [28]:

- Contradiction: $\perp \equiv \varphi^{\mathtt{M}} \wedge \neg\varphi^{\mathtt{M}}$,
- Tautology: $\top \equiv \neg\perp$,
- Or: $\varphi_1^{\mathtt{M}} \vee \varphi_2^{\mathtt{M}} \equiv \neg(\neg\varphi_1^{\mathtt{M}} \wedge \neg\varphi_2^{\mathtt{M}})$,
- Implication: $\varphi_1^{\mathtt{M}} \Rightarrow \varphi_2^{\mathtt{M}} \equiv \neg\varphi_1^{\mathtt{M}} \vee \varphi_2^{\mathtt{M}}$,
- **F**uture: $\mathbf{F}_I\varphi^{\mathtt{M}} \equiv \top\mathbf{U}_I\varphi^{\mathtt{M}}$,
- **G**lobally: $\mathbf{G}_I\varphi^{\mathtt{M}} \equiv \neg\mathbf{F}_I\neg\varphi^{\mathtt{M}}$,
- **O**nce: $\mathbf{O}_I\varphi^{\mathtt{M}} \equiv \top\mathbf{S}_I\varphi^{\mathtt{M}}$,
- **H**istorically: $\mathbf{H}_I\varphi^{\mathtt{M}} \equiv \neg\mathbf{O}_I\neg\varphi^{\mathtt{M}}$.

MTLp$_f$ over the *point-wise* semantics [46] is interpreted over timed traces, which can be thought of as sequences of events with timestamps. Given is a trace $\tau := (\tau^0, \dots, \tau^k, \dots)$ with length $|\tau|$, where $\tau^k : \mathcal{AP} \rightarrow \{\mathtt{true}, \mathtt{false}\}$ denotes a valuation over $\mathcal{AP}$, i.e., an assignment of `true` or `false` to every atomic proposition $\sigma \in \mathcal{AP}$, at step k. The notation $(\tau, k) \models \varphi^{\mathtt{M}}$ indicates that $\varphi^{\mathtt{M}}$ holds in the k-th valuation of τ, i.e., τ^k. We simplify the semantics of MTLp$_f$ in [28], [45] since valuations τ^k are synchronized with steps k:

- $(\tau, k) \models \sigma$ if and only if (iff) $\tau^k(\sigma) = \mathtt{true}$,
- $(\tau, k) \models \neg\varphi^{\mathtt{M}}$ iff $(\tau, k) \not\models \varphi^{\mathtt{M}}$,
- $(\tau, k) \models \varphi_1^{\mathtt{M}} \wedge \varphi_2^{\mathtt{M}}$ iff $(\tau, k) \models \varphi_1^{\mathtt{M}}$ and $(\tau, k) \models \varphi_2^{\mathtt{M}}$,
- $(\tau, k) \models \mathbf{X}_I\varphi^{\mathtt{M}}$ iff $k < |\tau|-1$, $1 \in I$, and $(\tau, k+1) \models \varphi^{\mathtt{M}}$,
- $(\tau, k) \models \mathbf{Y}_I\varphi^{\mathtt{M}}$ iff $k > 0$, $1 \in I$, and $(\tau, k-1) \models \varphi^{\mathtt{M}}$,
- $(\tau, k) \models \varphi_1^{\mathtt{M}}\mathbf{U}_I\varphi_2^{\mathtt{M}}$ iff $\exists l,\ k \leq l \leq |\tau| - 1$: $(\tau, l) \models \varphi_2^{\mathtt{M}}$, $l - k \in I$, and $\forall m,\ k \leq m < l$: $(\tau, m) \models \varphi_1^{\mathtt{M}}$,
- $(\tau, k) \models \varphi_1^{\mathtt{M}}\mathbf{S}\varphi_2^{\mathtt{M}}$ iff $\exists l,\ 0 \leq l \leq k$: $(\tau, l) \models \varphi_2^{\mathtt{M}}$, $k - l \in I$, and $\forall m,\ l < m \leq k$: $(\tau, m) \models \varphi_1^{\mathtt{M}}$.

As examples, formulas $\mathbf{X}_{[2,3]}\varphi^{\mathtt{M}}$ and $\varphi_1^{\mathtt{M}}\mathbf{U}_{[1,4]}\varphi_2^{\mathtt{M}}$ can be respectively read as "next valuation occurs within 2 and 3 steps (from now), in which $\varphi^{\mathtt{M}}$ holds" and "within 1 and 4 steps, a valuation occurs in which $\varphi_2^{\mathtt{M}}$ holds, and $\varphi_1^{\mathtt{M}}$ holds for all valuations before that". The past-time connectives **Y**, **S**, **O**, and **H** mirror their future-time counterparts **X**, **U**, **F**, and **G**, respectively, backward in time.

2) Linear Temporal Logic (with Past over Finite Traces): Since τ^k are synchronized with k, we do not require the full expressiveness of MTLp$_f$ for model checking our system. We interpret MTLp$_f$ formulas as *LTL with past over finite traces* (LTLp$_f$) [47] and further convert them into LTL over *infinite* traces for model checking. This reduces the complexity of model checking from EXPSPACE-complete for MTLp$_f$ [48] to PSPACE-complete for LTL [49]. Moreover, this allows us to employ mature and efficient LTL model checkers such as Spot [50]. MTLp$_f$ is syntactically reduced to LTLp$_f$ by dropping intervals I over the temporal connectives [47]; further dropping past-time connectives results in standard LTL [29]. We respectively denote by $\varphi^{\mathtt{L}}$, φ, and $\mathcal{F}$ an LTLp$_f$ formula, an LTL formula, and the set of formulas converted into LTL.

C. Set-based Reachability Analysis

Next, we define one-step reachable sets and drivable areas of the ego vehicle.

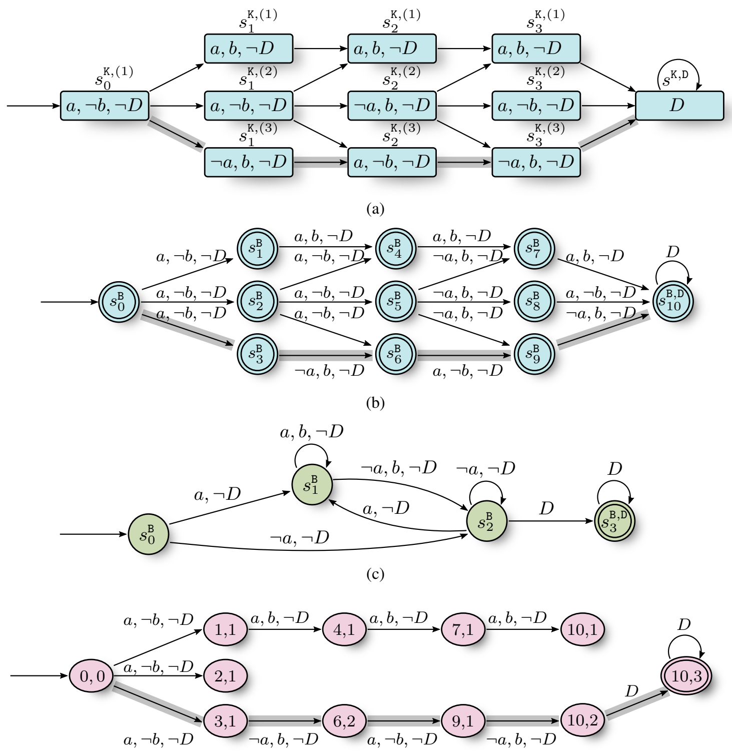
Definition 1 (Occupancy)**.** *The operator* $\mathrm{occ}(\cdot)$ *returns the occupied positions within* $F^{\mathtt{L}}$*. For example,* $\mathrm{occ}(\boldsymbol{x}_k)$ *returns the occupancy of the ego vehicle with state* $\boldsymbol{x}_k$*.*

Definition 2 (Set of forbidden states)**.** *Let* $\mathcal{O}_k \subset \mathbb{R}^2$ *be the set of positions occupied by all obstacles at step* k *and the space outside the road. The set of forbidden states of the ego vehicle at step* k *is defined as*

$$\mathcal{X}_k^{\mathtt{F}} := \left\{\boldsymbol{x}_k \in \mathcal{X}_k \,\middle|\, \mathrm{occ}(\boldsymbol{x}_k) \cap \mathcal{O}_k \neq \varnothing\right\}. \tag{2}$$

Definition 3 (One-Step Reachable Set)**.** *Let* $\mathcal{R}_0^{\mathtt{e}} = \mathcal{X}_0$ *be the exact reachable set of the ego vehicle at the initial step, with* $\mathcal{X}_0$ *being the set of collision-free initial states including measurement uncertainties. The exact reachable set* $\mathcal{R}_{k+1}^{\mathtt{e}}$ *is the set of states reachable from* $\mathcal{R}_k^{\mathtt{e}}$ *without intersecting the set of forbidden states* $\mathcal{X}_{k+1}^{\mathtt{F}}$*, denoted by* $\mathrm{reach}(\mathcal{R}_k^{\mathtt{e}})$*:*

$$\mathcal{R}_{k+1}^{\mathtt{e}} := \underbrace{\Big\{\boldsymbol{x}_{k+1} \in \mathcal{X}_{k+1} \Big| \exists \boldsymbol{x}_k \in \mathcal{R}_k^{\mathtt{e}}, \exists \boldsymbol{u}_k \in \mathcal{U}_k : \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k), \boldsymbol{x}_{k+1} \notin \mathcal{X}_{k+1}^{\mathtt{F}}\Big\}}_{\mathrm{reach}(\mathcal{R}_k^{\mathtt{e}})}. \tag{3}$$



$s_0^{\mathtt{K},(1)}$ $a, \neg b, \neg D$
$s_1^{\mathtt{K},(1)}$ $a, b, \neg D$
$s_2^{\mathtt{K},(1)}$ $a, b, \neg D$
$s_3^{\mathtt{K},(1)}$ $a, b, \neg D$
$s_1^{\mathtt{K},(2)}$ $a, \neg b, \neg D$
$s_2^{\mathtt{K},(2)}$ $\neg a, b, \neg D$
$s_3^{\mathtt{K},(2)}$ $a, \neg b, \neg D$
$s_1^{\mathtt{K},(3)}$ $\neg a, b, \neg D$
$s_2^{\mathtt{K},(3)}$ $a, \neg b, \neg D$
$s_3^{\mathtt{K},(3)}$ $\neg a, b, \neg D$
$s^{\mathtt{K},\mathtt{D}}$ D
(a)
$s_0^{\mathtt{B}}$ $s_1^{\mathtt{B}}$ $s_2^{\mathtt{B}}$ $s_3^{\mathtt{B}}$ $s_4^{\mathtt{B}}$ $s_5^{\mathtt{B}}$ $s_6^{\mathtt{B}}$ $s_7^{\mathtt{B}}$ $s_8^{\mathtt{B}}$ $s_9^{\mathtt{B}}$ $s_{10}^{\mathtt{B},\mathtt{D}}$
(b)
$a, b, \neg D$
$\neg a, b, \neg D$
$\neg a, \neg D$
$a, \neg D$
D
$s_0^{\mathtt{B}}$ $s_1^{\mathtt{B}}$ $s_2^{\mathtt{B}}$ $s_3^{\mathtt{B},\mathtt{D}}$
(c)
0, 0
1,1 4,1 7,1 10,1
2,1
3,1 6,2 9,1 10,2
10,3
(d)

Fig. 1: Minimal example of automata-based model checking. The colors correspond to components in Fig. 3. (a) A Kripke structure $G^{\mathtt{K}}$ with $\mathcal{AP} = \{a, b, D\}$, $\mathcal{S}^{\mathtt{K}} = \{s_0^{\mathtt{K},(1)}, \dots, s_3^{\mathtt{K},(3)}, s^{\mathtt{K},\mathtt{D}}\}$, and $\mathcal{S}_0^{\mathtt{K}} = \{s_0^{\mathtt{K},(1)}\}$. For clarity, we show all atomic propositions regardless of their truth values. (b) Automaton $A^{\mathtt{M}}$ converted from graph $G^{\mathtt{K}}$ in (a), with $\mathcal{S}^{\mathtt{B}} = \mathcal{S}^{\mathtt{BA}} = \{s_0^{\mathtt{B}}, \dots, s_9^{\mathtt{B}}, s_{10}^{\mathtt{B},\mathtt{D}}\}$. (c) Automaton A^{φ} converted from $\varphi := \mathbf{G}(a \Rightarrow \mathbf{X}(b))$: whenever a holds, b should hold in the next valuation. (d) Product automaton $A^{\mathtt{P}} = A^{\mathtt{M}} \otimes A^{\varphi}$. For brevity, we only show the subscripts of the states in $A^{\mathtt{P}}$. For example, the state 9,1 in $A^{\mathtt{P}}$ references $s_9^{\mathtt{B}}$ in $A^{\mathtt{M}}$ and $s_1^{\mathtt{B}}$ in A^{φ}. Transitions along the only accepting run in $A^{\mathtt{P}}$ are colored gray. For comparison, we also show other transitions that do not lead to an accepting state. The accepting run in $A^{\mathtt{P}}$ corresponds to the run $s_0^{\mathtt{B}} \to s_3^{\mathtt{B}} \to s_6^{\mathtt{B}} \to s_9^{\mathtt{B}} \to s_{10}^{\mathtt{B},\mathtt{D}} \to s_{10}^{\mathtt{B},\mathtt{D}} \to \dots$ in $A^{\mathtt{M}}$ and the execution $s_0^{\mathtt{K},(1)} \to s_1^{\mathtt{K},(3)} \to s_2^{\mathtt{K},(3)} \to s_3^{\mathtt{K},(3)} \to s^{\mathtt{K},\mathtt{D}} \to s^{\mathtt{K},\mathtt{D}} \to \dots$ in $G^{\mathtt{K}}$ (both colored gray). Please note that an auxiliary atomic proposition D (***Dead***) and an auxiliary self-looping state $s^{\mathtt{K},\mathtt{D}}$ (respectively $s^{\mathtt{B},\mathtt{D}}$) are required for extending the traces in $G^{\mathtt{K}}$ (respectively $A^{\mathtt{M}}$ and A^{φ}) to infinite length (see Sec. VI-3).

Definition 4 (Projection)**.** *The operator* $\mathrm{proj}_{\Diamond}(\boldsymbol{x})$ *maps the state* $\boldsymbol{x} \in \mathcal{X}$ *to its components* $\Diamond$*. For example,* $\mathrm{proj}_{(s,\dot{s})}(\boldsymbol{x}) = (s, \dot{s})^{\mathsf{T}}$ *for* $\boldsymbol{x} = (s, \dot{s}, \ddot{s})^{\mathsf{T}}$*. A set can be projected using the same operator:* $\mathrm{proj}_{\Diamond}(\mathcal{X}) = \{\mathrm{proj}_{\Diamond}(\boldsymbol{x}) \,|\, \boldsymbol{x} \in \mathcal{X}\}$.

Definition 5 (Drivable Area)**.** *The drivable area* $\mathcal{D}_k^{\mathtt{e}}$ *of the ego vehicle at step* k *is the projection of its reachable set* $\mathcal{R}_k^{\mathtt{e}}$ *onto the position domain:* $\mathcal{D}_k^{\mathtt{e}} := \mathrm{proj}_{(s,d)}(\mathcal{R}_k^{\mathtt{e}})$.

In practice, $\mathcal{X}_k^{\mathtt{F}}$ can be of arbitrary shape and the computation of $\mathcal{R}_k^{\mathtt{e}}$ as well as $\mathcal{D}_k^{\mathtt{e}}$ is generally difficult or even impossible [51]. Therefore, we compute their over-approximations $\mathcal{R}_k$ and $\mathcal{D}_k$, which will be detailed in Sec. IV.

D. Automata-based Model Checking

As motivated in Sec. I, we leverage model checking to efficiently and exhaustively identify all collision-free and specification-compliant driving corridors within the reachable sets of the ego vehicle. Let $A^{\mathtt{M}}$ be a finite state automaton representing a system M. To verify whether all possible executions of M satisfy a given LTL formula φ, denoted by $M \models \varphi$, the basic idea of automata-based model checking is to find a run in $A^{\mathtt{M}}$ that satisfies the negated formula $\neg\varphi$. If such a run does not exist, it can be concluded that $M \models \varphi$. Instead of examining whether $M \models \varphi$, model checking can alternatively be formulated to find the subset of runs in $A^{\mathtt{M}}$ that satisfy φ [52]. We follow the latter formulation since we aim to identify specification-compliant driving corridors rather than verifying whether all driving corridors satisfy the enforced specifications. We introduce two required definitions.

Definition 6 (Nondeterministic Büchi Automaton [53])**.** *A five-tuple* $(\Sigma, \mathcal{S}^{\mathtt{B}}, s_0^{\mathtt{B}}, \mathrm{trans}^{\mathtt{B}}, \mathcal{S}^{\mathtt{BA}})$ *defines a nondeterministic Büchi automaton, where*

- $\Sigma := \mathbb{P}(\mathcal{AP})^1$ *is an alphabet with letters* $\lambda \in \Sigma$,
- $\mathcal{S}^{\mathtt{B}}$ *is a set of states with elements* $s^{\mathtt{B}}$,

[1]The operator $\mathbb{P}(\cdot)$ returns the power set of the input.

- $s^{\mathtt{B}}_0 \in \mathcal{S}^{\mathtt{B}}$ *is the initial state,*
- $\mathrm{trans}^{\mathtt{B}} : \mathcal{S}^{\mathtt{B}} \times \Sigma \to \mathbb{P}(\mathcal{S}^{\mathtt{B}})$ *is a transition relation,*
- $\mathcal{S}^{\mathtt{BA}} \subseteq \mathcal{S}^{\mathtt{B}}$ *is a set of accepting states.*

A nondeterministic Büchi automaton is an finite state automaton accepting inputs of *infinite* length.

Definition 7 (Product Automaton [52])**.** *Given nondeterministic Büchi automata $A_m = (\Sigma, \mathcal{S}^{\mathtt{B}}_m, s^{\mathtt{B}}_{0,m}, \mathrm{trans}^{\mathtt{B}}_m, \mathcal{S}^{\mathtt{BA}}_m)$, $m \in \{1, 2\}$, their synchronous product is $A = A_1 \otimes A_2 := (\Sigma, \mathcal{S}^{\mathtt{P}}, s^{\mathtt{P}}_0, \mathrm{trans}^{\mathtt{P}}, \mathcal{S}^{\mathtt{PA}})$, where*

- $\mathcal{S}^{\mathtt{P}} = \mathcal{S}^{\mathtt{B}}_1 \times \mathcal{S}^{\mathtt{B}}_2$ *is the set of states with elements $(s^{\mathtt{B}}_1, s^{\mathtt{B}}_2)$,*
- $s^{\mathtt{P}}_0 = (s^{\mathtt{B}}_{0,1}, s^{\mathtt{B}}_{0,2}), s^{\mathtt{P}}_0 \in \mathcal{S}^{\mathtt{P}}$ *is the initial state,*
- $\mathrm{trans}^{\mathtt{P}} : \mathcal{S}^{\mathtt{P}} \times \Sigma \to \mathbb{P}(\mathcal{S}^{\mathtt{P}})$ *is a transition relation such that $(\tilde{s}^{\mathtt{B}}_1, \tilde{s}^{\mathtt{B}}_2) \in \mathrm{trans}^{\mathtt{P}}((s^{\mathtt{B}}_1, s^{\mathtt{B}}_2), \lambda)$ iff $\tilde{s}^{\mathtt{B}}_1 \in \mathrm{trans}^{\mathtt{B}}_1(s^{\mathtt{B}}_1, \lambda)$ and $\tilde{s}^{\mathtt{B}}_2 \in \mathrm{trans}^{\mathtt{B}}_2(s^{\mathtt{B}}_2, \lambda)$,*
- $\mathcal{S}^{\mathtt{PA}} \subseteq \mathcal{S}^{\mathtt{P}}$ *is a set of accepting states such that $(s^{\mathtt{B}}_1, s^{\mathtt{B}}_2) \in \mathcal{S}^{\mathtt{PA}}$ iff $s^{\mathtt{B}}_1 \in \mathcal{S}^{\mathtt{BA}}_1$ and $s^{\mathtt{B}}_2 \in \mathcal{S}^{\mathtt{BA}}_2$.*

Automaton A is also a nondeterministic Büchi automaton and accepts runs that are accepted by both automata A_1 and A_2. Fig. 1 depicts a minimal example of automata-based model checking, whose steps are presented as follows [52]:

1) Construct Automaton $A^{\mathtt{M}}$: Given a Kripke structure $G^{\mathtt{K}}$ (see Def. 11), it is converted into a nondeterministic Büchi automaton as described in [54]. Fig. 1a–b illustrate an exemplary $G^{\mathtt{K}}$ and the automaton $A^{\mathtt{M}}$ converted from $G^{\mathtt{K}}$.

2) Construct Automaton A^{φ}: An LTL formula φ can be readily translated into a Büchi automaton A^{φ} using, e.g., the tool Spot [50]. The reader is referred to [49], [50], [55] for further details. We use $\mathcal{A}^{\varphi} := \{\dots, A^{\varphi}_m, \dots\}$ to denote the set of automata converted from LTL formulas $\mathcal{F}$.

3) Retrieve Accepting Runs in Product Automaton $A^{\mathtt{P}}$: Let automaton $A^{\mathtt{P}}$ be the product of $A^{\mathtt{M}}$ and $\mathcal{A}^{\varphi}$ (see Sec. VII-A). Based on the Büchi acceptance condition [56], a run in a nondeterministic Büchi automaton is accepting if it visits some accepting states in $\mathcal{S}^{\mathtt{BA}}$ infinitely often. An accepting state is illustrated by a double circle (see Fig. 1b–d).

E. Driving Corridor

The reachable sets $\mathcal{R}_k$ of the ego vehicle enclose the collision-free solution space for motion planning; however, they may (a) be disconnected in the position domain due to the presence of obstacles and (b) contain states $\boldsymbol{x}_k$ having different valuations τ_k. This renders the direct usage of the reachable sets unsuitable for obtaining constraints for generating specification-compliant trajectories. To address this problem, we identify collision-free and specification-compliant *driving corridors* that are subsets of the reachable sets, which can be utilized as constraints over the states $\boldsymbol{x}_k$ in the motion planning problem. We present the necessary definitions.

Definition 8 (Connected Component)**.** *A connected component $\mathcal{C}_k \subseteq \mathcal{R}_k$ with valuation τ_k over $\mathcal{AP}$ is a set such that*

(a) $\mathcal{C}_k$ is a connected set [57] and collision-free in the position domain, i.e., $\mathcal{C}_k \cap \mathcal{X}^{\mathtt{F}}_k = \varnothing$,
(b) the states $\boldsymbol{x}_k$ in $\mathcal{C}_k$ have the same valuation τ_k.

Definition 9 (Driving Corridor)**.** *A driving corridor DC is a sequence of connected components $\mathcal{C}_k$ over steps 0 to k_h.*

Definition 10 (Specification-Compliant Driving Corridor)**.** *A driving corridor complying with specifications $\mathcal{F}$ is one such that $\forall \varphi \in \mathcal{F} : (\tau_0, \dots, \tau_{k_h}) \models \varphi$.*

F. Problem Statement

The problem we aim to solve is formally defined as follows:

Problem 1 (Optimal Specification-Compliant Driving Corridor Identification)**.** *The optimal specification-compliant driving corridor $DC^{\mathtt{O}}$ of the ego vehicle is one with the maximum utility over steps k:*

$$\max \sum_{k=0}^{k_h} u_k \tag{4a}$$

$$\text{subject to} \quad \boldsymbol{x}_0 \in \mathcal{C}_0, \tag{4b}$$

$$\forall k \in \{0, \dots, k_h - 1\} : \mathcal{C}_{k+1} \cap \mathrm{reach}(\mathcal{C}_k) \neq \varnothing, \tag{4c}$$

$$\forall \varphi \in \mathcal{F} : (\tau_0, \dots, \tau_{k_h}) \models \varphi, \tag{4d}$$

where u_k is the utility of $\mathcal{C}_k$ (see Sec. VII-C).

Constraints (4c) and (4d) respectively encode the reachability of successive connected components of $DC^{\mathtt{O}}$ and its compliance with the enforced specifications. Collision-freeness of $DC^{\mathtt{O}}$ follows directly from Def. 8. We aim to obtain $DC^{\mathtt{O}}$ and extract constraints over $\boldsymbol{x}_k$ for motion planning: Given a driving corridor, the motion planning problem can be formulated such that the trajectory of the ego vehicle is contained within the driving corridor.

Problem 2 (Motion Planning with Driving Corridor)**.** *Given a driving corridor, the motion planning problem is to minimize the cost function $J : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ over steps k:*

$$\min_{\boldsymbol{U}} \sum_{k=0}^{k_h} J(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{5a}$$

$$\text{subject to} \tag{5b}$$

$$\forall k \in \{0, \dots, k_h\} : \boldsymbol{x}_k \in \mathcal{C}_k, \tag{5c}$$

$$\forall k \in \{0, \dots, k_h - 1\} : \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k). \tag{5d}$$

III. Methodology

The input to our approach is the current environment model, including the road network, a curvilinear coordinate system $F^{\mathtt{L}}$, the set $\mathcal{F}$ of considered specifications, and all relevant obstacles, e.g., those perceived within a certain field of view of the ego vehicle. Without loss of generality, we assume the obstacles to be vehicles, each denoted by V_n. Furthermore, we assume that the predicted trajectories of all vehicles, e.g., their most likely trajectories, are given as input. For demonstration purposes, we consider interstate and intersection traffic rules formalized in [6], [7] as specifications. Nevertheless, our approach can be easily extended to handle other specifications expressible in MTLp$_f$, e.g., traffic rules described in [5], [8].

Let us introduce our approach for identifying collision-free and specification-compliant driving corridors, whose solution concept and relevant components are respectively illustrated in Fig. 2 and Fig. 3. As a first step, we apply reachability

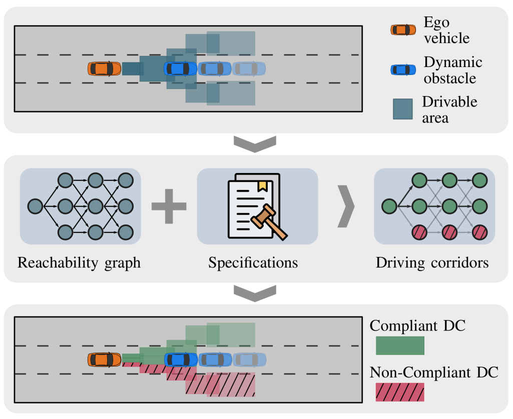


Fig. 2: Solution concept for identifying collision-free and specification-compliant driving corridors (DCs). Such driving corridors can be determined by model checking an automaton constructed from the reachability graph against automata translated from the enforced specifications. In this example, we assume overtaking from the right side is forbidden; thus, driving corridors corresponding to this maneuver are dismissed.

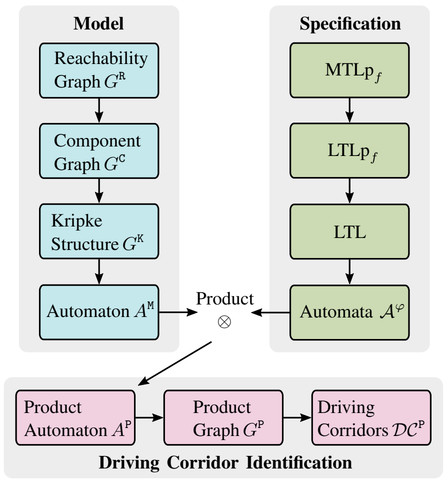


Fig. 3: Relationships of different components.

analysis on the ego vehicle to obtain its reachable set. Due to the presence of obstacles, the reachable set is represented as the union of multiple partial reachable sets, whose reachability and time relationships are stored in a *reachability graph* (see Sec. IV-A). Next, we process the reachability graph and the considered specifications for model checking:

1) As addressed in Sec. II-E, partial reachable sets may be disconnected or may contain states $\boldsymbol{x}_k$ having different valuations. To utilize their bounds as constraints for motion planning, they are grouped into *connected components*. These connected components form a *component graph*, based on which driving corridors are identified (see Sec. IV-B). A finite state automaton is constructed from the component graph and represents discrete state transitions of the ego vehicle over time (see Sec. II-D1).
2) The valuations of the partial reachable sets required for checking the compliance with specifications $\mathcal{F}$ are determined by evaluating a set of predicates (see Sec. V). Also, as motivated in Sec. II-B2, we rewrite MTLp$_f$ formulas as LTL formulas, whose details are presented in Sec. VI. The LTL formulas are translated into multiple finite state automata determining the accepting sequences of discrete state transitions (see Sec. II-D2).
3) As the number of driving corridors within a component graph grows exponentially with the planning horizon k_h, we rely on model checking to efficiently and exhaustively identify driving corridors complying with the enforced specifications $\mathcal{F}$. By computing the synchronous product of all automata, we obtain a product automaton based on which specification-compliant sequences of discrete state transitions and their corresponding driving corridors are identified (see Sec. II-D3).

Since numerous candidate driving corridors may exist, we generate a *product graph* from the product automaton, from which the optimal driving corridor is identified based on user-defined utilities (see Sec. VII). If the solution to (5) cannot be found within a driving corridor, we select the next optimal driving corridor. As long as time permits, trajectories can be planned for each available driving corridor; thus, it is possible to obtain multiple trajectory options.

The state and input of our vehicle model for reachability analysis only capture the position, velocity, and acceleration components of the ego vehicle (see Sec. IV); therefore, specifications concerning other components such as orientation and jerk cannot be handled using our approach. We resort to a trajectory repairer [58] to repair the planned trajectories so that the unconsidered specifications are also satisfied (whenever possible). If this also does not work, we execute a fail-safe trajectory as described in [59].

IV. Reachability Analysis

We describe the computation of reachability graphs based on [43] as well as its component graphs and driving corridors.

A. Reachability Graph

As motivated in Sec. II-C, we aim to compute the over-approximations of the exact reachable set $\mathcal{R}_k^{\mathsf{e}}$ and drivable area $\mathcal{D}_k^{\mathsf{e}}$ of the ego vehicle. For computational efficiency, the dynamics of the ego vehicle is abstracted by two double integrators within the coordinate system F^{L}, with the geometric center of the ego vehicle set as the reference point. The states and inputs in our model are $\boldsymbol{x}_k = (s_k, \dot{s}_k, d_k, \dot{d}_k)^\mathsf{T}$ and $\boldsymbol{u}_k = (\ddot{s}_k, \ddot{d}_k)^\mathsf{T}$, respectively:

$$\boldsymbol{x}_{k+1} = \underbrace{\begin{pmatrix} 1 & \Delta_t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta_t \\ 0 & 0 & 0 & 1 \end{pmatrix} \boldsymbol{x}_k + \begin{pmatrix} \frac{1}{2}\Delta_t^2 & 0 \\ \Delta_t & 0 \\ 0 & \frac{1}{2}\Delta_t^2 \\ 0 & \Delta_t \end{pmatrix} \boldsymbol{u}_k}_{f(\boldsymbol{x}_k, \boldsymbol{u}_k)}. \quad (6)$$

This abstraction ensures that the reachable sets of the adopted model always subsume those of high-fidelity vehicle models; alternative abstractions can be found in [60], [61]. Let $\square$ be a variable with its minimum and maximum values respectively denoted by $\underline{\square}$ and $\overline{\square}$. The velocities and accelerations at (s_k, d_k) are bounded by

$$\underline{\dot{s}}(\Gamma, s_k) \leq \dot{s}_k \leq \overline{\dot{s}}(\Gamma, s_k),\ \ \underline{\dot{d}}(\Gamma, s_k) \leq \dot{d}_k \leq \overline{\dot{d}}(\Gamma, s_k), \tag{7a}$$

$$\underline{\ddot{s}}(\Gamma, s_k) \leq \ddot{s}_k \leq \overline{\ddot{s}}(\Gamma, s_k),\ \ \underline{\ddot{d}}(\Gamma, s_k) \leq \ddot{d}_k \leq \overline{\ddot{d}}(\Gamma, s_k). \tag{7b}$$

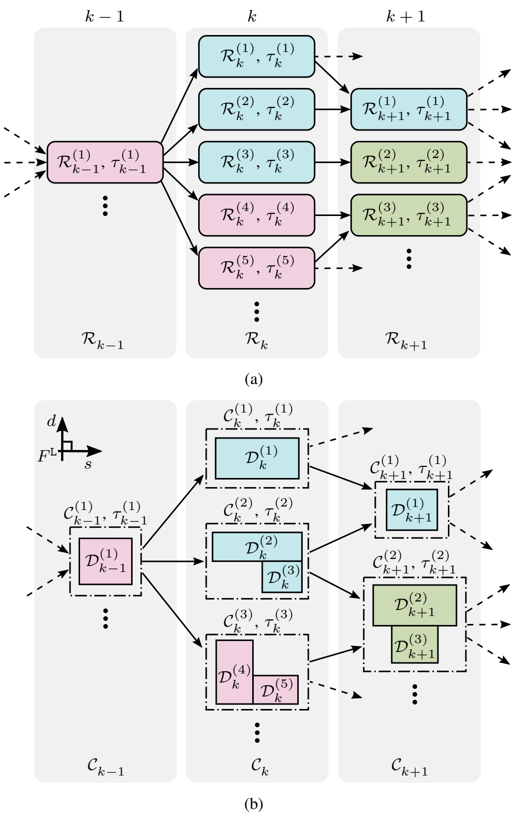
These bounds are chosen conservatively to consider the kinematic limitations within a curvilinear coordinate system, see, e.g., [62]. As a final check, the drivability of the planned trajectories should be examined separately, e.g., using the drivability checker described in [63].

Following [11], we under-approximate $\mathrm{occ}(\boldsymbol{x}_k)$ by its inscribed circle and $\mathcal{O}_k$ by axis-aligned rectangles accounting for its arbitrary shape, yielding an under-approximative set of forbidden states in (2). Therefore, the over-approximative reachable sets $\mathcal{R}_k \supseteq \mathcal{R}_k^{\mathtt{e}}$ enclose all drivable trajectories of the ego vehicle. To reduce computational complexity, we adopt the union of so-called *base sets* $\mathcal{R}_k^{(i)}$, $i \in \mathbb{N}$, as the set representation for $\mathcal{R}_k$, i.e., $\mathcal{R}_k := \cup_i \mathcal{R}_k^{(i)}$. Every base set $\mathcal{R}_k^{(i)}$ is a Cartesian product of two convex polytopes that enclose the reachable positions and velocities of the ego vehicle in the $(s, \dot{s})$ and $(d, \dot{d})$ planes, respectively. To simplify the notation, we also denote the collection[2] of $\mathcal{R}_k^{(i)}$ with $\mathcal{R}_k$, i.e., $\mathcal{R}_k := \{\ldots, \mathcal{R}_k^{(i)}, \ldots\}$. The unified valuation of the states $\boldsymbol{x}_k$ within $\mathcal{R}_k^{(i)}$ over atomic propositions $\mathcal{AP}$ is denoted by $\tau_k^{(i)}$. A directed and acyclic *reachability graph* $G^{\mathtt{R}}$ is computed as described in [43] to store the relationships of $\mathcal{R}_k^{(i)}$ in terms of reachability, see Fig. 4a. An edge $(\mathcal{R}_k^{(i)}, \mathcal{R}_{k+1}^{(j)})$ in graph $G^{\mathtt{R}}$ indicates that set $\mathcal{R}_{k+1}^{(j)}$ is reachable from set $\mathcal{R}_k^{(i)}$ after one step. Similar to Def. 5, the projections of $\mathcal{R}_k$ and $\mathcal{R}_k^{(i)}$ onto the position domain are respectively denoted by $\mathcal{D}_k$ and $\mathcal{D}_k^{(i)}$.

B. Component Graph and Driving Corridors

To facilitate the identification of driving corridors, we group the base sets $\mathcal{R}_k^{(i)}$ in a graph $G^{\mathtt{R}}$ into connected components $\mathcal{C}_k^{(j)}$, whose collection is denoted by $\mathcal{C}_k^{\mathtt{C}}$. Based on Def. 8, every connected component $\mathcal{C}_k^{(j)}$ with valuation $\tau_k^{(j)}$ over $\mathcal{AP}$ is a collection of base sets $\mathcal{R}_k^{(i)}$ such that (a) sets $\mathcal{R}_k^{(i)}$ form a connected set [57] and their drivable areas $\mathcal{D}_k^{(i)}$ are collision-free and (b) sets $\mathcal{R}_k^{(i)}$ and $\mathcal{C}_k^{(j)}$ have the same valuation, i.e., $\tau_k^{(j)} = \tau_k^{(i)}$. Without loss of generality, we assume that the set of initial states $\mathcal{X}_0$ of the ego vehicle is enclosed in the connected component $\mathcal{C}_0^{(1)}$. Connected components $\mathcal{C}_k^{(j)}$, together with edges connecting them, form a *component graph* $G^{\mathtt{C}}$, see Fig. 4b. An edge $(\mathcal{C}_k^{(j)}, \mathcal{C}_{k+1}^{(l)})$ in $G^{\mathtt{C}}$ indicates that at least one base set in $\mathcal{C}_k^{(j)}$ reaches a base set in $\mathcal{C}_{k+1}^{(l)}$ within one step. We also define the Kripke structure [64] from a graph $G^{\mathtt{C}}$, which is required for model checking, see Sec. II-D1. Fig. 1a shows an example of a Kripke structure $G^{\mathtt{K}}$.

[2]Throughout this article, a set of sets is referred to as a *collection*.



$k-1$
k
$k+1$
$\mathcal{R}_{k-1}^{(1)}, \tau_{k-1}^{(1)}$
$\mathcal{R}_k^{(1)}, \tau_k^{(1)}$
$\mathcal{R}_k^{(2)}, \tau_k^{(2)}$
$\mathcal{R}_k^{(3)}, \tau_k^{(3)}$
$\mathcal{R}_k^{(4)}, \tau_k^{(4)}$
$\mathcal{R}_k^{(5)}, \tau_k^{(5)}$
$\mathcal{R}_{k+1}^{(1)}, \tau_{k+1}^{(1)}$
$\mathcal{R}_{k+1}^{(2)}, \tau_{k+1}^{(2)}$
$\mathcal{R}_{k+1}^{(3)}, \tau_{k+1}^{(3)}$
$\mathcal{R}_{k-1}$
$\mathcal{R}_k$
$\mathcal{R}_{k+1}$
(a)
d
$F^{\mathtt{L}}$
s
$\mathcal{C}_{k-1}^{(1)}, \tau_{k-1}^{(1)}$
$\mathcal{D}_{k-1}^{(1)}$
$\mathcal{C}_k^{(1)}, \tau_k^{(1)}$
$\mathcal{D}_k^{(1)}$
$\mathcal{C}_k^{(2)}, \tau_k^{(2)}$
$\mathcal{D}_k^{(2)}$
$\mathcal{D}_k^{(3)}$
$\mathcal{C}_k^{(3)}, \tau_k^{(3)}$
$\mathcal{D}_k^{(4)}$
$\mathcal{D}_k^{(5)}$
$\mathcal{C}_{k+1}^{(1)}, \tau_{k+1}^{(1)}$
$\mathcal{D}_{k+1}^{(1)}$
$\mathcal{C}_{k+1}^{(2)}, \tau_{k+1}^{(2)}$
$\mathcal{D}_{k+1}^{(2)}$
$\mathcal{D}_{k+1}^{(3)}$
$\mathcal{C}_{k-1}$
$\mathcal{C}_k$
$\mathcal{C}_{k+1}$
(b)

Fig. 4: A reachability graph $G^{\mathtt{R}}$ and its component graph $G^{\mathtt{C}}$. Nodes of the same color have the same set of atomic propositions. (a) Graph $G^{\mathtt{R}}$ connecting nodes of different steps. (b) Graph $G^{\mathtt{C}}$ resulted from grouping the base sets $\mathcal{R}_k^{(i)}$ in $G^{\mathtt{R}}$ into connected components $\mathcal{C}_k^{(j)}$.

Definition 11 (Kripke Structure of Component Graph)**.** *The Kripke structure $G^{\mathtt{K}}$ of a component graph $G^{\mathtt{C}}$ is a four-tuple $(\mathcal{S}^{\mathtt{K}}, \mathcal{S}_0^{\mathtt{K}}, \mathrm{trans}^{\mathtt{K}}, \mathrm{label}^{\mathtt{K}})$:*

- *$\mathcal{S}^{\mathtt{K}} = \{\ldots, s_k^{\mathtt{K},(j)}, \ldots\} \cup \{s^{\mathtt{K,D}}\}$ is a set of states, where a state $s_k^{\mathtt{K},(j)}$ maps to a connected component $\mathcal{C}_k^{(j)}$ in $G^{\mathtt{C}}$; $s^{\mathtt{K,D}}$ is an auxiliary self-looping state required for extending a trace in $G^{\mathtt{K}}$ to infinite length.*
- *$\mathcal{S}_0^{\mathtt{K}} = \{s_0^{\mathtt{K},(1)}\}$ is a set of initial states.*
- *$\mathrm{trans}^{\mathtt{K}} : \mathcal{S}^{\mathtt{K}} \to \mathbb{P}(\mathcal{S}^{\mathtt{K}})$ is a transition relation and is defined as: $s_{k+1}^{\mathtt{K},(l)} \in \mathrm{trans}^{\mathtt{K}}(s_k^{\mathtt{K},(j)})$ if the edge $(\mathcal{C}_k^{(j)}, \mathcal{C}_{k+1}^{(l)})$ exists in $G^{\mathtt{C}}$; $s^{\mathtt{K,D}} \in \mathrm{trans}^{\mathtt{K}}(s_{k_h}^{\mathtt{K},(j)})$; $s^{\mathtt{K,D}} \in \mathrm{trans}^{\mathtt{K}}(s^{\mathtt{K,D}})$.*
- *$\mathrm{label}^{\mathtt{K}} : \mathcal{S}^{\mathtt{K}} \to \mathbb{P}(\mathcal{AP})$ is a labeling function that labels each state with a set of atomic propositions, which is defined as $\sigma \in \mathrm{label}(s_k^{\mathtt{K},(j)})$ if $\tau_k^{(j)}(\sigma) = \mathtt{true}$.*

Each path in a graph $G^{\mathtt{C}}$ corresponds to a collision-free driving corridor based on Def. 9. For example, let $k = 1$ and $k_h = 2$ in Fig. 4b. Sequences $DC_1 := (\mathcal{C}_0^{(1)}, \mathcal{C}_1^{(1)}, \mathcal{C}_2^{(1)})$ and $DC_2 := (\mathcal{C}_0^{(1)}, \mathcal{C}_1^{(3)}, \mathcal{C}_2^{(2)})$ correspond to two driving corridors of the ego vehicle. We utilize the position and velocity bounds

TABLE I: SELECTION OF CONSIDERED PREDICATES.

Category	**Type**	**Predicate**	**Rule** (see [6], [7])
Position	VI	in_lanelet, on_main_carriageway, behind_stop_line, at_traffic_sign, . . .	R-I5, R-IN1
	VD	in_front_of, behind, beside, left_of, right_of, in_same_lane, . . .	R-G1, R-I2
Velocity	VI	keeps_lane_speed_limit, perserves_flow, in_standstill, . . .	R-G3, R-G4, R-I1, R-IN2
	VD	keeps_safe_velocity_prec, drives_faster, . . .	R-G1, R-I2
Acceleration	VI	admissible_braking	R-G2
Priority	VD	has_priority_over, same_priority_as, . . .	R-IN3, R-IN4, R-IN5
Traffic Situation	VI	changes_lanelet, passing_stop_line, turning_left, turning_right, . . .	R-IN1, R-IN3
	VD	slow_leading_vehicle, in_congestion, cut_in, . . .	R-G1, R-G4, R-I1

of the connected component $\mathcal{C}_k^{(j)}$ within a driving corridor as constraints over the state $\boldsymbol{x}_k$ to restrict the planning space:

$$[\underline{\boldsymbol{x}}_k, \overline{\boldsymbol{x}}_k] = \text{hull}(\mathcal{C}_k^{(j)}), \tag{8}$$

where $\text{hull}(\cdot)$ returns the interval hull of a set.

V. Predicate Evaluation

The valuations over atomic propositions required for determining the satisfaction of specifications are often generated by evaluating a set of *predicates* formulated in higher-order logic. Our predicates have the general form of $\text{predicate}(\boldsymbol{x}_k; \cdot)$ and accept appropriate arguments. Tab. I lists selected predicates pertinent to rules formalized in [6], [7], which are divided into different categories and types. The evaluation of a *vehicle-dependent* (VD) predicate relies on other vehicles, whereas that of a *vehicle-independent* (VI) predicate does not.

Let us define some sets and functions to assist the evaluation of predicates. We denote by $\mathcal{L}$ the lanelets along the reference path Γ and their adjacent lanelets; the set $\mathcal{L}^{\texttt{dir}} \subset \mathcal{L}$ refers to lanelets having the same driving direction as the ego vehicle. The lanelets occupied by the ego vehicle with state $\boldsymbol{x}_k$ are obtained as follows:

$$\text{lanelets}(\boldsymbol{x}_k) := \left\{L \in \mathcal{L} \middle| \, \text{occ}(L) \cap \text{occ}(\boldsymbol{x}_k) \neq \varnothing \right\},$$
$$\text{lanelets\_dir}(\boldsymbol{x}_k) := \text{lanelets}(\boldsymbol{x}_k) \cap \mathcal{L}^{\texttt{dir}}.$$

The functions $\text{type}(L)$ and $\text{traffic\_sign}(L)$ return the type of a lanelet L (main carriageway, access ramp, etc.) and the set of traffic signs referenced by L, respectively. The functions $\text{front}(\cdot)$ and $\text{rear}(\cdot)$ return the s coordinate of the front and rear bumper of the input within F^{L}, respectively. Variable $\boldsymbol{x}_{n,k}^{\texttt{oth}}$ denotes the state of vehicle V_n at step k. We only describe a few exemplary predicates from each category for a concise presentation. The reader is referred to [6], [7] for detailed definitions of other predicates.

1) Position Predicates: Vehicle-independent position predicates relate to lanelets and traffic rule elements in the scenario. We provide three examples:

$$\begin{aligned}
&\text{in\_lanelet}(\boldsymbol{x}_k; L) \Leftrightarrow L \in \text{lanelets}(\boldsymbol{x}_k),\\
&\text{on\_main\_carriageway}(\boldsymbol{x}_k) \Leftrightarrow\\
&\qquad \text{main\_carriage\_way} \in \{\text{type}(L) | L \in \text{lanelets}(\boldsymbol{x}_k)\},\\
&\text{at\_traffic\_sign}(\boldsymbol{x}_k; TS) \Leftrightarrow\\
&\qquad \exists L \in \text{lanelets\_dir}(\boldsymbol{x}_k) : TS \in \text{traffic\_sign}(L),
\end{aligned}$$

where TS stands for a traffic sign. Vehicle-dependent position predicates reflect positional relations between the ego vehicle and other vehicles. For example, the mutually exclusive predicates $\text{in\_front\_of}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\texttt{oth}})$, $\text{behind}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\texttt{oth}})$, and $\text{beside}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\texttt{oth}})$ along the s direction can be evaluated with respect to V_n as follows:

$$\begin{aligned}
&\text{in\_front\_of}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\texttt{oth}}) \Leftrightarrow \text{rear}(\boldsymbol{x}_k) > \text{front}(\boldsymbol{x}_{n,k}^{\texttt{oth}}),\\
&\text{behind}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\texttt{oth}}) \Leftrightarrow \text{front}(\boldsymbol{x}_k) < \text{rear}(\boldsymbol{x}_{n,k}^{\texttt{oth}}),\\
&\text{beside}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\texttt{oth}}) \Leftrightarrow \big(\text{left\_of}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\texttt{oth}}) \vee \text{right\_of}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\texttt{oth}})\big)\\
&\wedge \neg\, \text{in\_front\_of}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\texttt{oth}}) \wedge \neg\, \text{behind}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\texttt{oth}}),
\end{aligned}$$

where the mutually exclusive predicates $\text{left\_of}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\texttt{oth}})$, $\text{right\_of}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\texttt{oth}})$, and $\text{aligned\_with}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\texttt{oth}})$ are analogously defined along the d direction.

2) Velocity Predicates: Vehicle-independent velocity predicates typically describe minimum or maximum velocity requirements. Rules R-G3 and R-G4 [7] specify different velocity limits that vehicles should respect, including limits introduced by the restricted field of view of a vehicle, the type of lane(let) in which the vehicle is driving, and the type of the vehicle. For instance, given the maximum velocity limit of the lane occupied with state $\boldsymbol{x}_k$, denoted by $\dot{s}^{\texttt{lane}}$, we have:

$$\text{keeps\_lane\_speed\_limit}(\boldsymbol{x}_k; \dot{s}^{\texttt{lane}}) \Leftrightarrow \text{proj}_{\dot{s}}(\boldsymbol{x}_k) \leq \dot{s}^{\texttt{lane}}.$$

Examples of vehicle-dependent velocity predicates indicate whether the ego vehicle is driving at a safe velocity with respect to a leading vehicle or driving faster than a vehicle. The latter predicate can be evaluated as follows:

$$\text{drives\_faster}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\texttt{oth}}) \Leftrightarrow \text{proj}_{\dot{s}}(\boldsymbol{x}_k) \geq \text{proj}_{\dot{s}}(\boldsymbol{x}_{n,k}^{\texttt{oth}}).$$

3) Acceleration Predicates: These predicates relate to the acceleration component of the ego vehicle. As an example, rule R-G2 [7] specifies situations in which a vehicle is allowed to brake harder than a predefined threshold. If the input $\boldsymbol{u}_k$ of the state $\boldsymbol{x}_k$ is within the range of admissible acceleration, the predicate $\text{admissible\_braking}(\boldsymbol{x}_k)$ evaluates to `true`.

4) Priority Predicates: Vehicles should respect driving priorities specified by traffic regulations, which can be inferred from the road structure or indicated by traffic signs. The predicates $\text{has\_priority\_over}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\texttt{oth}})$ and $\text{same\_priority\_as}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\texttt{oth}})$ reflect whether the ego vehicle has priority over V_n or has the same priority as V_n, respectively. They are evaluated by comparing the driving priorities determined based on the current traffic scenario and road priorities listed in [6, Tab. II].

5) Traffic Situation Predicates: The truth of these predicates depends on given traffic situations. For example, vehicle-independent predicates may indicate whether the ego vehicle is passing a stop line:

$$\begin{aligned}&\text{passing\_stop\_line}(\boldsymbol{x}_k) \Leftrightarrow\\ &\qquad \text{behind\_stop\_line}(\boldsymbol{x}_k) \wedge \mathbf{X}(\neg\, \text{behind\_stop\_line}(\boldsymbol{x}_k)),\end{aligned}$$

where behind_stop_line($\boldsymbol{x}_k$) evaluates to `true` if the ego vehicle is behind a stop line. Vehicle-dependent predicates may indicate whether a slow leading vehicle exists and whether a vehicle is stuck in traffic congestion.

VI. Specification Rewriting

As motivated in Sec. II-B2, we rewrite and interpret an MTLp$_f$ formula $\varphi^{\mathtt{M}}$ as an LTL formula φ on our system:

1) Eliminate Intervals over Temporal Connectives: Since valuations τ_k of our traces are synchronized with steps k, we rewrite an MTLp$_f$ connective as a combination of $\mathbf{X}$ and $\mathbf{Y}$ connectives in LTLp$_f$. We use the notation $\mathbf{X}_{[\tilde{k}]}$ as a shorthand for $\tilde{k}$ consecutive $\mathbf{X}$ connectives:

$$\mathbf{X}_{[\tilde{k}]}\varphi^{\mathtt{L}} := \underbrace{\mathbf{X}\mathbf{X}\ldots\mathbf{X}}_{\tilde{k}\text{ times }\mathbf{X}}\varphi^{\mathtt{L}}. \tag{9}$$

It follows from the semantics of $\mathbf{X}_I$, $\mathbf{U}_I$ (see Sec. II-B1) that the time interval $I = [a, b]$ over future-time connectives can be eliminated:

$$\mathbf{X}_{[a,b]}\varphi^{\mathtt{L}} = \begin{cases}\mathbf{X}\varphi^{\mathtt{L}}, & \text{if } 1 \in [a,b].\\ \perp, & \text{otherwise.}\end{cases} \tag{10}$$

$$\varphi_1^{\mathtt{L}}\mathbf{U}_{[a,b]}\varphi_2^{\mathtt{L}} = \bigvee_{a\le\tilde{k}\le b}\big(\mathbf{G}_{[0,\tilde{k}-1]}\varphi_1^{\mathtt{L}} \wedge \mathbf{X}_{[\tilde{k}]}\varphi_2^{\mathtt{L}}\big), \tag{11}$$

$$\mathbf{G}_{[a,b]}\varphi^{\mathtt{L}} = \bigwedge_{a\le\tilde{k}\le b}\mathbf{X}_{[\tilde{k}]}\varphi^{\mathtt{L}}, \tag{12}$$

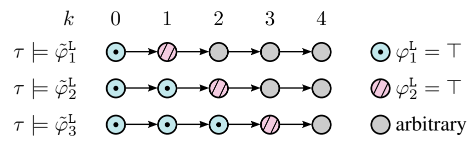
$$\mathbf{F}_{[a,b]}\varphi^{\mathtt{L}} = \bigvee_{a\le\tilde{k}\le b}\mathbf{X}_{[\tilde{k}]}\varphi^{\mathtt{L}}. \tag{13}$$

That is, $\mathbf{X}_{[a,b]}\varphi^{\mathtt{L}}$ is only satisfiable by τ if the unit step jump is within $[a, b]$ and $\varphi^{\mathtt{L}}$ holds in the next valuation; $\varphi_1^{\mathtt{L}}\mathbf{U}_{[a,b]}\varphi_2^{\mathtt{L}}$ is satisfied if within a and b steps, a valuation occurs in which $\varphi_2^{\mathtt{L}}$ holds, and $\varphi_1^{\mathtt{L}}$ continuously holds for valuations before that; $\mathbf{G}_{[a,b]}\varphi^{\mathtt{L}}$ ($\mathbf{F}_{[a,b]}\varphi^{\mathtt{L}}$) is satisfied if $\varphi^{\mathtt{L}}$ holds in all (any) valuations occurring within a and b steps. Intervals over the past-time connectives $\mathbf{Y}_I$, $\mathbf{O}_I$, $\mathbf{H}_I$, and $\mathbf{S}_I$ can be analogously eliminated by rewriting using the $\mathbf{Y}$ connective.

Running example:

$$\begin{aligned}\varphi_1^{\mathtt{L}}\mathbf{U}_{[1,3]}\varphi_2^{\mathtt{L}} &\overset{(11)}{=} \tilde{\varphi}_1^{\mathtt{L}} \vee \tilde{\varphi}_2^{\mathtt{L}} \vee \tilde{\varphi}_3^{\mathtt{L}}, \text{where}\\ \tilde{\varphi}_1^{\mathtt{L}} &:= \mathbf{G}_{[0,0]}\varphi_1^{\mathtt{L}} \wedge \mathbf{X}_{[1]}\varphi_2^{\mathtt{L}} \overset{(12)}{=} \varphi_1^{\mathtt{L}} \wedge \mathbf{X}\varphi_2^{\mathtt{L}},\\ \tilde{\varphi}_2^{\mathtt{L}} &:= \mathbf{G}_{[0,1]}\varphi_1^{\mathtt{L}} \wedge \mathbf{X}_{[2]}\varphi_2^{\mathtt{L}} \overset{(12)}{=} \varphi_1^{\mathtt{L}} \wedge \mathbf{X}\varphi_1^{\mathtt{L}} \wedge \mathbf{X}_{[2]}\varphi_2^{\mathtt{L}},\\ \tilde{\varphi}_3^{\mathtt{L}} &:= \mathbf{G}_{[0,2]}\varphi_1^{\mathtt{L}} \wedge \mathbf{X}_{[3]}\varphi_2^{\mathtt{L}}\\ &\overset{(12)}{=} \varphi_1^{\mathtt{L}} \wedge \mathbf{X}\varphi_1^{\mathtt{L}} \wedge \mathbf{X}_{[2]}\varphi_1^{\mathtt{L}} \wedge \mathbf{X}_{[3]}\varphi_2^{\mathtt{L}}.\end{aligned}$$

Fig. 5 shows traces satisfying $\tilde{\varphi}_1^{\mathtt{L}}$, $\tilde{\varphi}_2^{\mathtt{L}}$, and $\tilde{\varphi}_3^{\mathtt{L}}$.



k
0
1
2
3
4
$\tau \models \tilde{\varphi}_1^{\mathtt{L}}$
$\tau \models \tilde{\varphi}_2^{\mathtt{L}}$
$\tau \models \tilde{\varphi}_3^{\mathtt{L}}$
$\varphi_1^{\mathtt{L}} = \top$
$\varphi_2^{\mathtt{L}} = \top$
arbitrary

Fig. 5: Example traces satisfying $\tilde{\varphi}_1^{\mathtt{L}}$, $\tilde{\varphi}_2^{\mathtt{L}}$, and $\tilde{\varphi}_3^{\mathtt{L}}$, respectively. $\tilde{\varphi}_1^{\mathtt{L}} := \mathbf{G}_{[0,0]}\varphi_1^{\mathtt{L}} \wedge \mathbf{X}_{[1]}\varphi_2^{\mathtt{L}}$, $\tilde{\varphi}_2^{\mathtt{L}} := \mathbf{G}_{[0,1]}\varphi_1^{\mathtt{L}} \wedge \mathbf{X}_{[2]}\varphi_2^{\mathtt{L}}$, $\tilde{\varphi}_3^{\mathtt{L}} := \mathbf{G}_{[0,2]}\varphi_1^{\mathtt{L}} \wedge \mathbf{X}_{[3]}\varphi_2^{\mathtt{L}}$. A circle represents a valuation in which the atomic proposition corresponding to the color is assigned `true`. A sequence of circles represents a trace.

2) Eliminate Past-Time Connectives: A syntactic procedure for separating past-time and future-time connectives in LTL is presented in [65], which has been further applied to LTLp$_f$ in [47]. Although the procedure offers straightforward rules for rewriting, it leads to so-called non-elementary blow-up in formula size [66], [67]. As an alternative, one can explicitly reformulate $\varphi^{\mathtt{L}}$ using only future-time connectives with an exponential growth in the formula size [68]. For practical reasons, we adopt a less strict rewriting procedure to avoid the mentioned unfavorable complexities. To this end, we temporarily switch from the *strong* semantics defined in Sec. II-B1 to the *repeat* semantics as described in [69] for the $\mathbf{X}$ and $\mathbf{Y}$ connectives, which allows one to cancel out pairs of $\mathbf{X}$ and $\mathbf{Y}$: Intuitively, we expect that the *previous* step of the *next* step along a trace τ is the *current* step, and vice versa:

$$\mathbf{X}\mathbf{Y}\varphi^{\mathtt{L}} \Rightarrow \varphi^{\mathtt{L}}, \tag{14}$$

$$\mathbf{Y}\mathbf{X}\varphi^{\mathtt{L}} \Rightarrow \varphi^{\mathtt{L}}. \tag{15}$$

After canceling out all pairs of $\mathbf{X}$ and $\mathbf{Y}$, we restore the *strong* semantics for interpreting the remaining $\mathbf{Y}\varphi^{\mathtt{L}}$, i.e., they all evaluate to $\perp$: $\mathbf{Y}\varphi^{\mathtt{L}}$ asserts that there exists a valuation prior to τ^0 and $\varphi^{\mathtt{L}}$ is true therein, which does not hold since our traces start with τ^0 at step $k = 0$. As for the $\mathbf{S}$ connective, we apply the axiom [70, A12]

$$\varphi_1^{\mathtt{L}}\mathbf{S}\varphi_2^{\mathtt{L}} = \varphi_2^{\mathtt{L}} \vee (\varphi_1^{\mathtt{L}} \wedge \mathbf{Y}(\varphi_1^{\mathtt{L}}\mathbf{S}\varphi_2^{\mathtt{L}})) \tag{16}$$

and examine the expanded formula.

Running example:

$$\begin{aligned}\mathbf{F}_{[0,2]}(\varphi_1^{\mathtt{L}}\mathbf{S}\varphi_2^{\mathtt{L}}) &\overset{(13)}{=} \tilde{\varphi}_1^{\mathtt{L}} \vee \tilde{\varphi}_2^{\mathtt{L}} \vee \tilde{\varphi}_3^{\mathtt{L}}, \text{where}\\ \tilde{\varphi}_1^{\mathtt{L}} &:= \varphi_1^{\mathtt{L}}\mathbf{S}\varphi_2^{\mathtt{L}}\\ &\overset{(16)}{=} \varphi_2^{\mathtt{L}} \vee (\varphi_1^{\mathtt{L}} \wedge \mathbf{Y}(\varphi_1^{\mathtt{L}}\mathbf{S}\varphi_2^{\mathtt{L}})) = \varphi_2^{\mathtt{L}},\\ \tilde{\varphi}_2^{\mathtt{L}} &:= \mathbf{X}_{[1]}(\varphi_1^{\mathtt{L}}\mathbf{S}\varphi_2^{\mathtt{L}})\\ &\overset{(16)}{=} \mathbf{X}(\varphi_2^{\mathtt{L}} \vee (\varphi_1^{\mathtt{L}} \wedge \mathbf{Y}(\varphi_1^{\mathtt{L}}\mathbf{S}\varphi_2^{\mathtt{L}})))\\ &= \mathbf{X}\varphi_2^{\mathtt{L}} \vee (\mathbf{X}\varphi_1^{\mathtt{L}} \wedge (\underline{\varphi_1^{\mathtt{L}}\mathbf{S}\varphi_2^{\mathtt{L}}}))\\ &\overset{\tilde{\varphi}_1^{\mathtt{L}}}{=} \mathbf{X}\varphi_2^{\mathtt{L}} \vee (\mathbf{X}\varphi_1^{\mathtt{L}} \wedge \underline{\varphi_2^{\mathtt{L}}}),\\ \tilde{\varphi}_3^{\mathtt{L}} &:= \mathbf{X}_{[2]}(\varphi_1^{\mathtt{L}}\mathbf{S}\varphi_2^{\mathtt{L}})\\ &\overset{(16)}{=} \mathbf{X}_{[2]}(\varphi_2^{\mathtt{L}} \vee (\varphi_1^{\mathtt{L}} \wedge \mathbf{Y}(\varphi_1^{\mathtt{L}}\mathbf{S}\varphi_2^{\mathtt{L}})))\\ &= \mathbf{X}_{[2]}\varphi_2^{\mathtt{L}} \vee (\mathbf{X}_{[2]}\varphi_1^{\mathtt{L}} \wedge \underline{\mathbf{X}(\varphi_1^{\mathtt{L}}\mathbf{S}\varphi_2^{\mathtt{L}})})\\ &\overset{\tilde{\varphi}_2^{\mathtt{L}}}{=} \mathbf{X}_{[2]}\varphi_2^{\mathtt{L}} \vee\\ &\quad (\mathbf{X}_{[2]}\varphi_1^{\mathtt{L}} \wedge (\underline{\mathbf{X}\varphi_2^{\mathtt{L}} \vee (\mathbf{X}\varphi_1^{\mathtt{L}} \wedge \varphi_2^{\mathtt{L}})})).\end{aligned}$$

Fig. 6 shows traces satisfying $\tilde{\varphi}_1^{\mathtt{L}}$, $\tilde{\varphi}_2^{\mathtt{L}}$, and $\tilde{\varphi}_3^{\mathtt{L}}$.

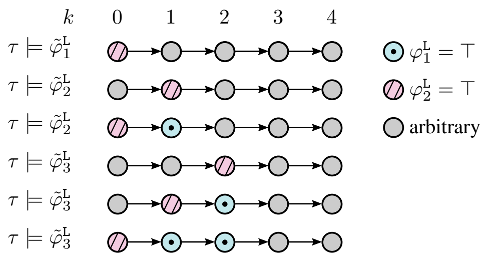

k
0
1
2
3
4
$\tau \models \tilde{\varphi}_1^{\mathtt{L}}$
$\tau \models \tilde{\varphi}_2^{\mathtt{L}}$
$\tau \models \tilde{\varphi}_2^{\mathtt{L}}$
$\tau \models \tilde{\varphi}_3^{\mathtt{L}}$
$\tau \models \tilde{\varphi}_3^{\mathtt{L}}$
$\tau \models \tilde{\varphi}_3^{\mathtt{L}}$
$\varphi_1^{\mathtt{L}} = \top$
$\varphi_2^{\mathtt{L}} = \top$
arbitrary

Fig. 6: Example traces satisfying $\tilde{\varphi}_1^{\mathtt{L}}$, $\tilde{\varphi}_2^{\mathtt{L}}$, and $\tilde{\varphi}_3^{\mathtt{L}}$, respectively. $\tilde{\varphi}_1^{\mathtt{L}} := \varphi_2^{\mathtt{L}}$, $\tilde{\varphi}_2^{\mathtt{L}} := \mathbf{X}\varphi_2^{\mathtt{L}} \vee (\mathbf{X}\varphi_1^{\mathtt{L}} \wedge \varphi_2^{\mathtt{L}})$, $\tilde{\varphi}_3^{\mathtt{L}} := \mathbf{X}_{[2]}\varphi_2^{\mathtt{L}} \vee (\mathbf{X}_{[2]}\varphi_1^{\mathtt{L}} \wedge (\mathbf{X}\varphi_2^{\mathtt{L}} \vee (\mathbf{X}\varphi_1^{\mathtt{L}} \wedge \varphi_2^{\mathtt{L}})))$.

3) Conversion to LTL: An LTLp$_f$ formula without past-time connectives is converted to an LTL formula φ based on [71], [72]. This conversion introduces an auxiliary atomic proposition D (***Dead***) in φ and an auxiliary self-looping state $s^{\mathtt{B,D}}$ in the automaton A^{φ} translated from φ (see Fig. 1c).

VII. Driving Corridor Identification

This section describes the computation of automaton $A^{\mathtt{P}}$ and the generation of its product graph, based on which we identify (optimal) specification-compliant driving corridors.

A. Product Automaton Computation

Given an automaton $A^{\mathtt{M}}$ and a set of automata $\mathcal{A}^{\varphi}$ (see Sec. II-D2), their product $A^{\mathtt{P}}$ can be computed considering factors such as flexibility and priorities in case the specifications are conflicting, i.e., cannot be satisfied by any trace. Rulebooks [73] specify qualitative relations between specifications as a pre-order and assigns the same priority to specifications in a group. Following this concept, automaton $A^{\mathtt{P}}$ can be computed to expedite compliance with the groups of specifications of higher priorities:

$$A_0^{\mathtt{P}} = A^{\mathtt{M}},\ A_{\tilde{m}}^{\mathtt{P}} = A_{\tilde{m}-1}^{\mathtt{P}} \otimes \underbrace{(\cdots \otimes A_m^{\varphi} \otimes \ldots)}_{\tilde{A}_{\tilde{m}}^{\varphi}}, \tag{17}$$

where $\tilde{A}_{\tilde{m}}^{\varphi}$ denotes the product of the automata in a group with priority $\tilde{m}$ (a smaller $\tilde{m}$ indicates a higher priority). Automaton $A_{\tilde{m}-1}^{\mathtt{P}}$ is assigned to $A^{\mathtt{P}}$ if an accepting run exists in $A_{\tilde{m}-1}^{\mathtt{P}}$ but not in $A_{\tilde{m}}^{\mathtt{P}}$. Two special instances with drawbacks exist that should ideally be avoided:

1) Assigning the Same Priority to All Automata A_m^{φ}: This instance yields an exponential growth in the number of states in $\tilde{A}_{\tilde{m}}^{\varphi}$ with respect to $|\mathcal{A}^{\varphi}|$ (see Def. 7). Moreover, it does not allow one to flexibly adjust enforced specifications per the current traffic situation or their orders based on user-defined measures such as importance or criticality. The latter property is unfavorable when not all prescribed specifications can be satisfied: possible reasons are conflicts in the specifications, misbehavior of other vehicles, etc.

2) Assigning a Unique Priority to Each Automaton A_m^{φ}: This instance allows one to explicitly prioritize the specifications and expedite compliance with those of higher priorities; however, meticulously ordering specifications becomes non-trivial as $|\mathcal{A}^{\varphi}|$ increases.

Algorithm 1 Remove Unreachable Base Sets

Inputs: Collections $\mathcal{C}_k^{\mathtt{C}}$ of connected components $\mathcal{C}_k^{(j)}$.
Output: Updated connected components $\mathcal{C}_k^{(j)}$.

```
 1: $\mathcal{R}_0^{\mathtt{keep}} \leftarrow \mathcal{C}_0^{(1)}$.BaseSets()                         ▷ Initialization
 2: for $k = 1$ to $k_h$ do
 3:     $\mathcal{R}_k^{\mathtt{keep}} \leftarrow \varnothing$                    ▷ Collection of base sets to keep at $k$
 4:     for $\mathcal{C}_k^{(j)} \in \mathcal{C}_k^{\mathtt{C}}$ do
 5:         $\mathcal{R}^{\mathtt{C}} \leftarrow \mathcal{C}_k^{(j)}$.BaseSets()      ▷ Base sets to keep in $\mathcal{C}_k^{(j)}$
 6:         for $\mathcal{R}_k^{(i)} \in \mathcal{C}_k^{(j)}$.BaseSets() do
 7:             if $\mathcal{R}_k^{(i)}$.ParentBaseSets() $\cap\, \mathcal{R}_{k-1}^{\mathtt{keep}} = \varnothing$ then
 8:                 $\mathcal{R}^{\mathtt{C}} \leftarrow \mathcal{R}^{\mathtt{C}} \setminus \{\mathcal{R}_k^{(i)}\}$              ▷ Remove $\mathcal{R}_k^{(i)}$
 9:             else
10:                 $\mathcal{R}_k^{\mathtt{keep}} \leftarrow \mathcal{R}_k^{\mathtt{keep}} \cup \{\mathcal{R}_k^{(i)}\}$        ▷ Keep $\mathcal{R}_k^{(i)}$ at $k$
11:             end if
12:         end for
13:         $\mathcal{C}_k^{(j)}$.BaseSets() $\leftarrow \mathcal{R}^{\mathtt{C}}$       ▷ Update base sets in $\mathcal{C}_k^{(j)}$
14:     end for
15: end for
```

B. Product Graph Generation

Given an automaton $A^{\mathtt{P}}$ with at least an accepting run, we convert it into a directed, acyclic, and weighted graph $G^{\mathtt{P}}$, which is referred to as a *product graph*. Graph $G^{\mathtt{P}}$ retains the general structure of $A^{\mathtt{P}}$ and consists of nodes referencing corresponding connected components $\mathcal{C}_k^{(j)}$. States in $A^{\mathtt{P}}$ with outgoing edges for which the auxiliary atomic proposition D is assigned `true` are dismissed in $G^{\mathtt{P}}$ since they are irrelevant to the identification of driving corridors, see Fig. 7. Every edge $(\mathcal{C}_{k-1}^{(l)}, \mathcal{C}_k^{(j)})$ in $G^{\mathtt{P}}$ is weighted by the utility of $\mathcal{C}_k^{(j)}$, denoted by $u_k^{(j)}$ (detailed in Sec. VII-C). The paths in $G^{\mathtt{P}}$ from $\mathcal{C}_0^{(1)}$ to $\mathcal{C}_{k_h}^{(j)}$ correspond to specification-compliant driving corridors (see Def. 9 and Def. 10) and are stored in a collection $\mathcal{DC}^{\mathtt{P}}$.

Let $\mathcal{C}^{\mathtt{P}}$ and $\mathcal{C}^{\mathtt{C}}$ represent the collection of all connected components in graphs $G^{\mathtt{P}}$ and $G^{\mathtt{C}}$, respectively. Since $\mathcal{C}^{\mathtt{P}} \subseteq \mathcal{C}^{\mathtt{C}}$, we update the reachability relationship between the base sets $\mathcal{R}_k^{(i)}$ in the connected components and by this remove $\mathcal{R}_k^{(i)}$ that no longer have a valid parent. For example, suppose $DC_1 := (\mathcal{C}_0^{(1)}, \mathcal{C}_1^{(2)}, \mathcal{C}_2^{(2)})$ in Fig. 4b is the only path in $G^{\mathtt{P}}$, set $\mathcal{R}_2^{(3)}$ is no longer reachable along DC_1 as per Fig. 4a. Alg. 1 removes unreachable base sets from connected components: For every step k, we maintain a collection $\mathcal{R}_k^{\mathtt{keep}}$ of base sets to be kept, with $\mathcal{R}_0^{\mathtt{keep}}$ initialized with the base sets in $\mathcal{C}_0^{(1)}$ (Alg. 1, line 1). For steps 1 to k_h, we iterate through $\mathcal{C}_k^{(j)} \in \mathcal{C}_k^{\mathtt{C}}$ and examine each of its base sets $\mathcal{R}_k^{(i)}$. If none of the parent base sets of $\mathcal{R}_k^{(i)}$ is present in $\mathcal{R}_{k-1}^{\mathtt{keep}}$, $\mathcal{R}_k^{(i)}$ is removed from $\mathcal{C}_k^{(j)}$; otherwise it is added to $\mathcal{R}_k^{\mathtt{keep}}$ (Alg. 1, lines 7–11).

C. Utility Computation

Identifying the optimal specification-compliant driving corridor, i.e., the solution to Prob. 1, requires computing the utility $u_k^{(j)}$ of connected components $\mathcal{C}_k^{(j)}$. Since multiple base sets $\mathcal{R}_k^{(i)}$ may exist in a connected component $\mathcal{C}_k^{(j)}$, we define a function $\mathrm{w\_mean}(\mathcal{C}_k^{(j)}, \Diamond)$ that returns the weighted mean of

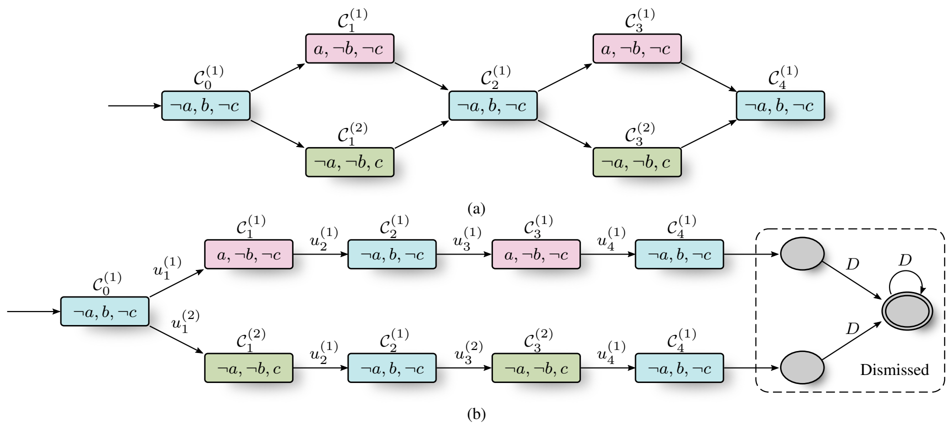


Fig. 7: Example of a component graph $G^{\mathtt{C}}$ and a product graph $G^{\mathtt{P}}$. Nodes of the same color have the same set of true atomic propositions. (a) Graph $G^{\mathtt{C}}$ with $k_h = 4$. (b) Graph $G^{\mathtt{P}}$ converted from a product automaton $A^{\mathtt{P}}$, which is the output of model checking $G^{\mathtt{C}}$ against specification $\varphi := \mathbf{G}(\neg a) \vee \mathbf{G}(\neg c)$: either a never holds, or c never holds. Dismissed states in $A^{\mathtt{P}}$ are shown in gray (cf. Fig. 1d).

component $\Diamond$ in $\mathcal{C}_k^{(j)}$:

$$\mathrm{w\_mean}(\mathcal{C}_k^{(j)}, \Diamond) := \sum_{\mathcal{R}_k^{(i)} \in \mathcal{C}_k^{(j)}} w_k^{(i)} \, \mathrm{mean}\big(\mathrm{proj}_{(\Diamond)}(\mathcal{R}_k^{(i)})\big), \tag{18}$$

$$w_k^{(i)} := \frac{\mathrm{area}(\mathcal{R}_k^{(i)})}{\sum_{\mathcal{R}_k^{(l)} \in \mathcal{C}_k^{(j)}} \mathrm{area}(\mathcal{R}_k^{(l)})}, \tag{19}$$

where $w_k^{(i)}$ is the weight of $\mathcal{R}_k^{(i)}$ within $\mathcal{C}_k^{(j)}$ and $\mathrm{area}(\cdot)$ returns the area of the input in the position domain. The utility $u_k^{(j)}$ of $\mathcal{C}_k^{(j)}$ is defined as the weighted sum of partial utilities:

$$u_k^{(j)} := \boldsymbol{w}^\top \boldsymbol{u}_k^{(j)}, \tag{20}$$

where $\boldsymbol{w}$ is a weighting vector and $\boldsymbol{u}_k^{(j)}$ is a vector of user-defined partial utilities. We consider the following partial utilities, which are all normalized to $[0, 1]$:

1) Area: We reward $\mathcal{C}_k^{(j)}$ of a larger area in the position domain since this generally yields more flexible position constraints for subsequent trajectory planning:

$$u^{\mathtt{area}}(\mathcal{C}_k^{(j)}) := \frac{\mathrm{area}(\mathcal{C}_k^{(j)})}{\max_{\mathcal{C}_k^{(l)} \in \mathcal{C}^{\mathtt{P}}} \mathrm{area}\, \mathcal{C}_k^{(l)}}. \tag{21}$$

2) Velocity: We reward $\mathcal{C}_k^{(j)}$ of higher weighted longitudinal velocity to increase the traffic flow:

$$u^{\mathtt{vel}}(\mathcal{C}_k^{(j)}) := \frac{\mathrm{w\_mean}(\mathcal{C}_k^{(j)}, \dot{s}) - \dot{s}_0}{\overline{\ddot{s}}(\Gamma, s_k)\, \Delta_t\, k}. \tag{22}$$

3) Position: We encourage $\mathcal{C}_k^{(j)}$ of longer weighted traveled distance in the longitudinal direction of the reference path:

$$u^{\mathtt{pos}}(\mathcal{C}_k^{(j)}) := \frac{\mathrm{w\_mean}(\mathcal{C}_k^{(j)}, s) - s_0}{0.5\, \overline{\ddot{s}}(\Gamma, s_k)\, (\Delta_t\, k)^2 + \dot{s}_0\, \Delta_t\, k}, \tag{23}$$

4) Reference Path: We penalize $\mathcal{C}_k^{(j)}$ of larger weighted lateral deviation from the reference path:

$$u^{\mathtt{ref}}(\mathcal{C}_k^{(j)}) := \exp\big(-w^{\mathtt{ref}}\ \mathrm{w\_mean}(\mathcal{C}_k^{(j)}, d)\big), \tag{24}$$

where $w^{\mathtt{ref}} \in \mathbb{R}_+$ is a factor dictating how fast $u^{\mathtt{ref}}(\mathcal{C}_k^{(j)})$ approaches zero as the lateral deviation increases. Alternative utilities such as comfort, criticality measures, and robustness degrees of specifications can be taken into consideration, whose computation is out of the scope of this article.

D. Optimal Driving Corridor

Given a graph $G^{\mathtt{P}}$, graph-search and sampling-based techniques can be employed to extract optimal paths in $G^{\mathtt{P}}$ with respect to $u_k^{(j)}$. For instance, the longest paths from the root node $\mathcal{C}_0^{(1)}$ to nodes $\mathcal{C}_{k_h}^{(j)}$ can be efficiently obtained using a single-source shortest path algorithm on graph $-G^{\mathtt{P}}$ in which the weights are negated [74]. These paths correspond to collision-free and specification-compliant driving corridors with the maximum cumulative weights and are stored in the collection $\mathcal{DC}^{\mathtt{O}}$. Every candidate in $\mathcal{DC}^{\mathtt{O}}$ is processed again using Alg. 1 to remove unreachable base sets. We identify the optimal driving corridor $DC^{\mathtt{O}}$ with the highest cumulative weight, within which trajectories are planned.

VIII. Evaluation

This section evaluates our approach and demonstrates its applicability, effectiveness, and efficiency. To this end, we integrate identified driving corridors into two sampling-based motion planners and compare the planning results under different traffic scenarios and specifications. In addition, we evaluate the performance of our approach under increasingly critical scenarios and compare computation times. Furthermore, we benchmark of computation time of our prototype against multiple scenarios. Lastly, we compare our approach with that described in [19].

TABLE II: Selected Parameters Used in the Experiments

Parameter	k_h	Δ_t	$\underline{\dot{s}}$	$\overline{\dot{s}}$	$\underline{\dot{d}}$	$\overline{\dot{d}}$	$\underline{\ddot{s}}$	$\overline{\ddot{s}}$	$\underline{\ddot{d}}$	$\overline{\ddot{d}}$
Value	15	0.2	0.0	20.0	−4.0	4.0	−6.0	6.0	−2.0	2.0

A. Implementation Details

For evaluation, we adopt scenarios from the CommonRoad benchmark suite[3] [75], whose typical components are a road network consisting of lanelets, static and dynamic obstacles, traffic rule elements such as traffic signs and traffic lights, the initial state of the ego vehicle, and a goal region. Every scenario has a unique benchmark ID and can be unambiguously reproduced. The prototype of our approach extends [76] and is partially implemented in Python and C++. We ran the experiments on a laptop with an Intel Core i7-7700HQ 2.8GHz processor. Tab. II lists selected parameters. The weights in $\boldsymbol{w}$ for driving corridor identification are all empirically set to 1.0. We briefly introduce the two adopted motion planners:

1) Reactive Planner: The popular motion planner described in [77], which we refer to as the reactive planner, generates a finite set of candidate trajectories connecting the initial state of the ego vehicle to different goal states. These goal states are generated based on samples of longitudinal velocity, lateral position, and the terminal time of the lateral maneuver. The candidate trajectories are checked for (a) feasibility (including drivability and collisions) using the drivability checker in [63] and (b) compliance with specifications using Spot [50].

2) RRT:* To showcase the possibility of integrating our approach with RRT-based planners that we reviewed in Sec. I-A, we also consider the RRT* planner [41]. In our implementation, a tree is incrementally constructed from sampled nodes, between which a trajectory is generated using Dubins car model [78]. As with the reactive planner, we check the feasibility and compliance with the specifications of the trajectories and terminate once a solution is found.

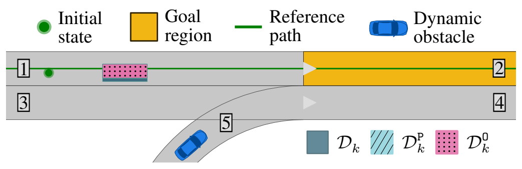
Besides these planners, it has been shown in [12], [13] that optimization-based planners also substantially benefit from the integration of driving corridors.

B. Scenario I: Merging via On-Ramp

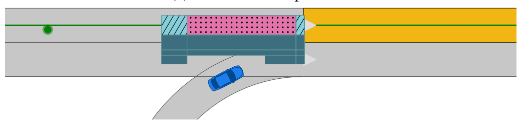
Fig. 8 depicts a baseline scenario where the ego vehicle is driving on a two-lane main carriageway and another vehicle is approaching via an on-ramp. While the ego vehicle is dynamically able to proceed in its current lane or to change to the lane on the right, rule R-I5 [7] prohibits the latter maneuver as the ego vehicle has to respect entering vehicles:

$$
\begin{aligned}
\mathbf{G}\Big(\big(&\text{on\_main\_carriageway}(\boldsymbol{x}_k) \land \text{behind}(\boldsymbol{x}_k; \boldsymbol{x}^{\mathtt{oth}}_{n,k}) \land \\
&\text{on\_access\_ramp}(\boldsymbol{x}^{\mathtt{oth}}_{n,k}) \land \\
&\mathbf{F}\big(\text{on\_main\_carriageway}(\boldsymbol{x}^{\mathtt{oth}}_{n,k})\big)\big) \\
&\Rightarrow \\
&\big(\text{on\_main\_carriageway\_right\_lane}(\boldsymbol{x}_k) \lor \\
&\mathbf{G}\big(\neg\,\text{on\_main\_carriageway\_right\_lane}(\boldsymbol{x}_k)\big)\big)\Big).
\end{aligned}
$$

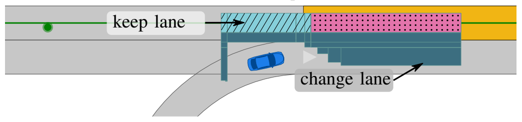
[3]http://commonroad.in.tum.de/



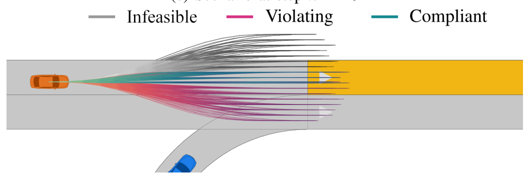
(a) Scenario at step $k = 5$



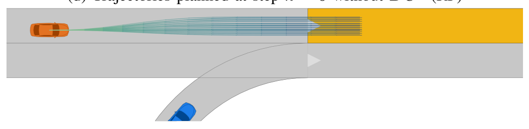
(b) Scenario at step $k = 10$



(c) Scenario at step $k = 15$



(d) Trajectories planned at step $k = 0$ without DC^0 (RP)



(e) Trajectories planned at step $k = 0$ with DC^0 (RP+DC/RP+DC*)

Fig. 8: Drivable areas and planned trajectories for scenario I (benchmark ID: `ZAM_TIV-1_1_T-1`).

In addition to the baseline scenario, we create two alternative scenarios with increased difficulty by adding secondary specifications: in variant 1, we require that the ego vehicle reaches lanelet 2 before the end of the planning horizon; in variant 2, lanelet 2 should be reached between steps 5 and 12, which is a stricter requirement with a smaller solution space.

Fig. 8a–c visualize the computed drivable areas at different steps. Because the connected components in $G^{\mathtt{P}}$ reference a subset of base sets in $G^{\mathtt{R}}$, their drivable areas at step k, denoted by $\mathcal{D}^{\mathtt{P}}_k$, is a subset of $\mathcal{D}_k$. Furthermore, the drivable areas of the optimal driving corridor DC^0 at step k, represented by $\mathcal{D}^0_k$, is a subset of $\mathcal{D}^{\mathtt{P}}_k$ since DC^0 corresponds to a path in $G^{\mathtt{P}}$. That is, $\mathcal{D}^0_k \subseteq \mathcal{D}^{\mathtt{P}}_k \subseteq \mathcal{D}_k$. The non-empty drivable area $\mathcal{D}^0_k$ implies that one may find a specification-compliant trajectory within the position and velocity bounds extracted from DC^0. We generate three sets of trajectories using the reactive planner under three settings:

- RP: the basic implementation of the reactive planner with fixed sampling intervals [77].
- RP+DC: enhances RP by drawing time, position, and

velocity samples within DC^0 as described in [14].

- RP+DC*: in addition to [14], enforces constraint (5c) and discards a trajectory if any of its states is outside DC^0.

Fig. 8d–e illustrate the sampled trajectories under different settings. While the trajectories sampled with RP covers both lanes and even off-road region, those planned with RP+DC/RP+DC* lie within the current lane of the ego vehicle.

Tab. III reports the computation results, among which we focus on the feasible trajectories and their compliance rate. In the baseline scenario, while around 40% of the trajectories planned with RP violate rule R-IN5 by entering the lane on the right, all trajectories considering DC^0 comply with the rule. In both alternative scenarios, the compliance rate of the trajectories sampled with RP significantly decreases, with that drastically reduced to around 0.5% in variant 2. Although the compliance rate of RP+DC exhibits a milder drop than that of RP, it is less than ideal because not all states of the planned trajectories are entirely contained in DC^0. In contrast, RP+DC* performed consistently well in the given scenarios. Further enforcing a conflicting or non-satisfiable specification (e.g., $\mathbf{F}_{[0,5]}\big(\text{in\_lanelet}(L_2)\big)$, see Fig. 8a) would yield an empty product graph G^{P}; thus, no DC^0 would be output and we can reject the specification before trying to plan a trajectory satisfying the specification.

We also compare the time required to obtain the first specification-compliant trajectory under different settings. Since the trajectory sampling and the feasibility check are shared among all three settings, we focus on generating a compliant trajectory from the feasible candidates. The computation is repeated for 50 times and the candidate trajectories are shuffled in each iteration. For RP+DC and RP+DC*, we also include the computation time of reachable sets. Fig. 9 depicts the computation results: RP required less (median) computation time than the other two settings in the baseline scenario and variant 1. This is justified by the fact that the enforced specifications in these two scenarios are relatively easy to be satisfied by the ego vehicle. With increased difficulty in variant 2, the computation time of RP grows remarkably (almost two orders of magnitude) since it struggles to find the few compliant trajectories among a large number of candidates. In contrast, the computation times of the settings adopting our reachable sets are consistent across the scenarios regardless of the considered specifications. In variant 2, the overhead of our reachable set computation is compensated by restricting the sampling space and, with RP+DC*, avoiding excessive compliance checks for sampled trajectories. Although RP+DC and RP+DC* performed similarly in these scenarios, adopting the latter allows us to explicitly constrain the sampled trajectories to the optimal driving corridor concerning user-defined utilities presented in Sec. VII-C.

C. Scenario II: Four-way Intersection

Next, we consider a scenario in which the ego vehicle must come to a full stop before an intersection to respect passing priorities. Rule R-IN3 [6] dictates that the ego vehicle should not endanger another entering vehicle at an intersection if it is left of the other vehicle. Due to space limitations, we refer the reader to [6] for the MTL formulation of this rule. We also create an alternative scenario to increase the difficulty of the planning problem. Specifically, we alter the initial velocity of the ego vehicle from $7.0\,\mathrm{m/s}$ to $9.0\,\mathrm{m/s}$, which reduces the compliant drivable areas and state space.

TABLE III: Number of trajectories and compliance rate under different settings in scenario I: `ZAM_TIV-1_1_T-1`.

Method	#Sampled	#Feasible	#In Corridor	#Compliant	%Compliant
Baseline: R-IN5					
RP	13464	3839	-	2232	58.14 %
RP+DC	11288	4031	-	4031	100.00 %
RP+DC*	11288	4031	64	64	100.00 %
Variant 1: R-IN5 + $\mathbf{F}_{[0,k_h]}\big(\text{in\_lanelet}(L_2)\big)$					
RP	13464	3839	-	968	25.21 %
RP+DC	11288	4031	-	3418	84.79 %
RP+DC*	11288	4031	62	62	100.00 %
Variant 2: R-IN5 + $\mathbf{F}_{[5,12]}\big(\text{in\_lanelet}(L_2)\big)$					
RP	13464	3839	-	18	0.47 %
RP+DC	11288	4031	-	1021	25.33 %
RP+DC*	11288	4031	62	62	100.00 %

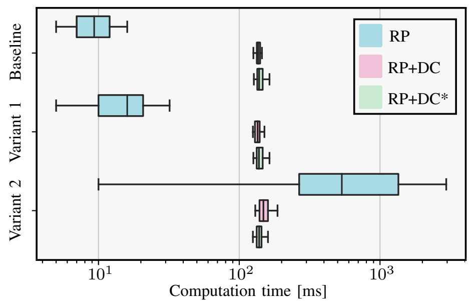


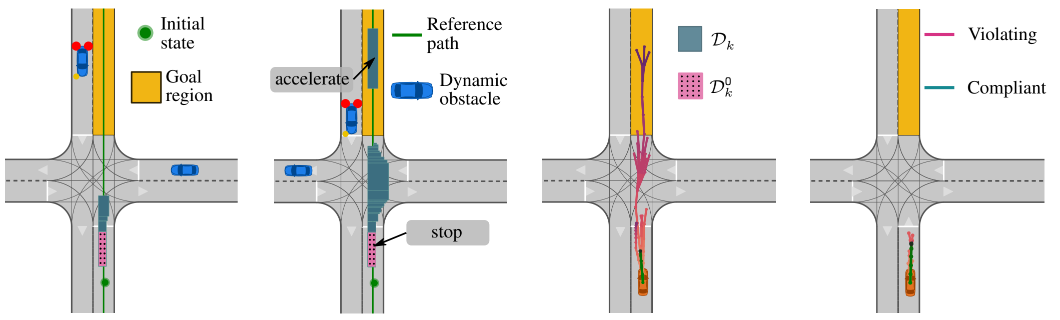
Fig. 9: Computation times in scenario I: `ZAM_TIV-1_1_T-1`. For better visibility, outliers of the box plot are not shown.

Fig. 10a–b illustrate the drivable areas of the ego vehicle in the baseline scenario. The ego vehicle can, among other maneuvers, accelerate and pass through the intersection before the vehicle entering from the right or respect the passing priority and stop before the intersection. The optimal driving corridor DC^0 is the only path in the product graph G^{P}, thus $\mathcal{D}^{\mathsf{P}}_k = \mathcal{D}^0_k$. We demonstrate the benefits of our approach for RRT-based planners by comparing the following settings:

- RRT*: the basic implementation of RRT*. For a fairer comparison, we restrict the state and sample spaces to lanelets on the route leading to the goal region.
- RRT*+DC*: based on RRT*, we enforce constraint (5c) by restricting the state and sample spaces to DC^0.

Fig. 10c–d show exemplary explored trees under different settings. While the tree explored by RRT* spans to incoming and outgoing lanelets of the intersection, the tree explored by RRT*+DC* is, as expected, contained within the incoming lanelet before the intersection.

We compare the number of tree nodes required to generate a collision-free and specification-compliant trajectory lasting 3.0 seconds. To account for the stochastic nature of RRT*, we ran the planners for 50 times and present the results in Fig. 11. For



(a) Scenario at step $k = 7$ (b) Scenario at step $k = 15$ (c) Trajectories planned at step $k = 0$ without DC^0 (RRT*) (d) Trajectories planned at step $k = 0$ with DC^0 (RRT*+DC*)

Fig. 10: Drivable areas and planned trajectories for scenario II (benchmark ID: `ZAM_TIV-2_1_T-1`).

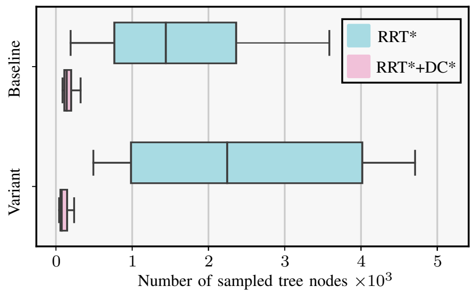


Fig. 11: Number of sampled tree nodes before finding a compliant trajectory in scenario II: `ZAM_TIV-2_1_T-1`. For better visibility, outliers of the box plot are not shown.

both the baseline and variant scenarios, the median numbers of sampled tree nodes of RRT*+DC* are substantially lower than those of RRT*. This can be explained by the fact that only a fraction of the state and sample spaces are relevant for planning a trajectory satisfying rule R-IN3. Reducing the specification-compliant drivable areas noticeably increases the effort for planning a compliant trajectory by RRT*, which is not the case for RRT*+DC*. These observations are in line with our findings in Sec. VIII-B.

D. Scenarios with Decreasing Solution Spaces

It has been demonstrated in [12, Sec. VII-E] that the computation times of reachable sets are proportionally reduced with a decreasing solution space. We verify the validity of this finding on our reachable set computation by considering a cluttered scenario populated with vehicles and cyclists, see Fig. 12. To decrease the solution space, we gradually raise the initial velocity of the ego vehicle by $30\,\%$ at a time until a collision is unavoidable. This process increases the criticality of the scenario based on measures such as Time-to-Collision and Time-to-React, which can be evaluated using the CriMe toolbox [79]. We repeat the computations for 50 times and list the results in Tab. IV. Increasing the initial velocity of the ego vehicle leads to reduced numbers of base sets in the reachability graph and required set operations, resulting in lower mean computation times and smaller overall sizes of the drivable area cumulated over steps k. We observe that with the initial velocity raised to $310\,\%$, which yields the most critical scenario with inevitable collision using parameters in Tab. II, the computation time is exceptionally low, at approximately $4\,\mathrm{ms}$. The results thus confirm the favorable property of our reachable set that less computation time is required in more critical scenarios with smaller solution spaces.

E. Computation Time

The performance of our prototype is benchmarked by computing the reachable sets for over 50 randomly chosen scenarios from the CommonRoad benchmark suite. We only focus on position predicates concerning lanelets and vehicles as well as traffic situation predicates. The former causes frequent splitting of reachable sets and the latter requires relatively more effort in the annotation operation [43]. Fig. 13 illustrates the computation times of required operations in our reachable set computation as described in [43, Sec. III-D]. Our current implementation, with 75% of the computations executed within $250\,\mathrm{ms}$, requires only a fraction of the planning horizon, specifically $3.0\,\mathrm{s}$, thereby demonstrating its real-time capability. To further improve the performance of our prototype, adequate optimization and parallelized computation techniques can be employed. For instance, our observations from [76] suggest that translating the annotation operation from Python to C++ is expected to accelerate its computation by a factor of 20. For both scenarios I and II presented in the previous subsections, computing the product automaton A^{P} and determining the optimal driving corridor DC^0 required only about $100\,\mu\mathrm{s}$ and $1\,\mathrm{ms}$, respectively.

F. Comparison

While most of the works that we reviewed in Sec. I-A focus on reach-avoid problems with temporal requirements for robot navigation, to the best of our knowledge, article [19] is the only work that aims to achieve a goal similar to ours. Specifically, the article (a) focuses on constraint extraction for motion

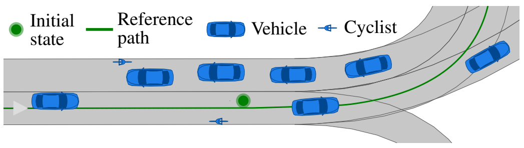


Fig. 12: A cluttered scenario with vehicles and cyclists at step $k = 0$ (benchmark ID: `ESP_Monzon-2_2_T-1`).

TABLE IV: Mean computation time, number of base sets, and size of the drivable area (in relative percentage) cumulated over steps k, when increasing the initial velocity in scenario `ESP_Monzon-2_2_T-1`.

Init. Vel.	%Vel.	Comput. Time	#Base Sets	%Drivable Area
4.69 m/s	100 %	139.51 ms	165	100.00 %
6.10 m/s	130 %	126.87 ms	160	99.33 %
7.51 m/s	160 %	124.47 ms	155	97.44 %
8.92 m/s	190 %	113.07 ms	148	96.06 %
10.32 m/s	220 %	98.10 ms	134	86.30 %
11.73 m/s	250 %	84.60 ms	118	71.67 %
13.14 m/s	280 %	58.30 ms	83	52.24 %
14.55 m/s	310 %	3.88 ms	5	0.13 %

planning of automated vehicles, (b) considers compliance with specifications in temporal logic, and (c) handles dynamic obstacles. For this reason, we compare our approach to [19].

Let us recapitulate the approach presented in [19]: The authors first partition the collision-free state space and construct a so-called *navigation graph* $G^{\mathtt{N}}$, in which a node denotes a segment of a lanelet with a unique position relation concerning other vehicles. Connecting such nodes forms a path representing a timed *envelope*, i.e., position constraints, enclosing a set of homotopic trajectories. Next, to examine the compliance of these envelopes with traffic rules expressed in LTL, each path in $G^{\mathtt{N}}$ is individually verified using runtime verification. Finally, the authors assign heuristic costs to specification-compliant envelopes, from which the best solutions are output as constraints for trajectory planning.

Our approach outweighs [19] in the following two aspects:

1) Model Accuracy: Article [19] does not incorporate a vehicle model accounting for the dynamics of the ego vehicle and only constructs $G^{\mathtt{N}}$ at the sub-lanelet level. In contrast, our approach adopts a double-integrator point mass model (6), effectively capturing the ego vehicle's position, velocity, and acceleration components. While both approaches extract position constraints for the ego vehicle that comply with enforced specifications, our driving corridors additionally offer velocity constraints. This allows us to integrate specifications pertinent to the velocity of the ego vehicle (see Tab. I). Also, specifications on the accelerations can be handled directly during our computation of reachable sets through the modification of input bounds (7b).

In addition, our approach provides a less over-approximative abstraction of the ego vehicle. This can be substantiated by comparing the sizes of the discrete system models in the two approaches. For comparison, we consider a scenario (benchmark ID: `ZAM_TIV-3_1_T-1`) featuring three parallel lanelets, each containing two other vehicles. Due to the limitations of [19], only the position predicates relative to other vehicles are considered in the comparison. Using the setting

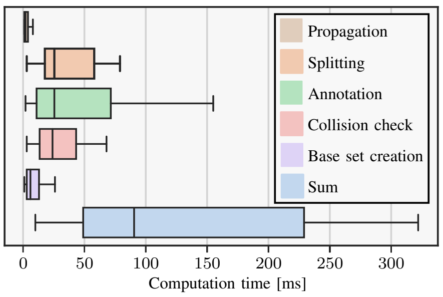


Fig. 13: Benchmarked computation times of our reachable set computation. For better visibility, outliers of the box plot are not shown.

described in Sec. VIII-A, our approach generates a component graph $G^{\mathtt{C}}$ comprised of nearly 180 nodes, which is significantly less than about 520 nodes in graph $G^{\mathtt{N}}$.

2) Verification Efficiency: Since the number of possible paths in graphs $G^{\mathtt{N}}$ and $G^{\mathtt{C}}$ grows exponentially in relation to the planning horizon k_h, even for the relatively simple scenario described in Sec. VIII-F1, graph $G^{\mathtt{N}}$ already contains about 250 billion paths to be monitored. This task is computationally demanding, if not intractable, for motion planning of automated vehicles with strict real-time requirements. Moreover, the task is incomplete unless all paths are examined. In stark contrast, our employment of automata-based model checking ensures that all paths in graph $G^{\mathtt{C}}$ are efficiently verified. At the same time, the computational complexity only increases linearly with the number of nodes in graph $G^{\mathtt{C}}$ [49], thereby demonstrating a far superior efficiency compared to runtime verification adopted by [19].

IX. Conclusions

Our novel approach offers a promising solution to the problem of specification-compliant motion planning for automated vehicles, paving the way to safer and more efficient road traffic. By coupling set-based reachability analysis with automata-based model checking, we identify collision-free and specification-compliant driving corridors of the ego vehicle. The driving corridors can be integrated into arbitrary motion planners accepting position and velocity constraints to expedite the generation of specification-compliant trajectories. In contrast to existing works, our approach realizes exhaustive verification of all possible driving corridors of the ego vehicle while accounting for its system dynamics and not sacrificing real-time capability. Moreover, the generation of a product graph enables detecting conflicting or non-satisfiable specifications before actually planning a trajectory. The experiments show that our approach can be easily integrated into motion planners to efficiently obtain trajectories complying with temporal specifications, especially when the solution space is increasingly small. Although our computation of reachable sets requires only a fraction of time of the planning horizon, as demonstrated with benchmarking over 50 CommonRoad scenarios, we will further improve the implementation so that it can achieve even better run time.

Acknowledgment

This work was funded by the German Research Foundation (DFG) under grant No. AL 1185/20-1, the German Federal Ministry for Education and Research (BMBF) under grant No. 03ZU1105KA (MCube), and Huawei Technologies under grant No. YBN2020035151. The authors also appreciate the fruitful collaboration with the project partners.

References

[1] S. Aradi, "Survey of deep reinforcement learning for motion planning of autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 740–759, 2020.

[2] L. Claussmann, M. Revilloud, D. Gruyer, and S. Glaser, "A review of motion planning for highway autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 5, pp. 1826–1848, 2019.

[3] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, 2016.

[4] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 1135–1145, 2015.

[5] Y. Sun, C. M. Poskitt, J. Sun, Y. Chen, and Z. Yang, "LawBreaker: An approach for specifying traffic laws and fuzzing autonomous vehicles," in *Profc. of the IEEE/ACM Int. Conf. Autom. Software Eng.*, 2022, pp. 1–12.

[6] S. Maierhofer, P. Moosbrugger, and M. Althoff, "Formalization of intersection traffic rules in temporal logic," in *Proc. of the IEEE Intell. Veh. Symp.*, 2022, pp. 1135–1144.

[7] S. Maierhofer, A.-K. Rettinger, E. C. Mayer, and M. Althoff, "Formalization of interstate traffic rules in temporal logic," in *Proc. of the IEEE Intell. Veh. Symp.*, 2020, pp. 752–759.

[8] K. Esterle, L. Gressenbuch, and A. Knoll, "Formalizing traffic rules for machine interpretability," in *Proc. of the IEEE Connect. Autom. Veh. Symp.*, 2020, pp. 1–7.

[9] A. Rizaldi, J. Keinholz, M. Huber, J. Feldle, F. Immler, M. Althoff, E. Hilgendorf, and T. Nipkow, "Formalising and monitoring traffic rules for autonomous vehicles in Isabelle/HOL," in *Int. Conf. Integr. Formal Methods*, 2017, pp. 50–66.

[10] E. Plaku and S. Karaman, "Motion planning with temporal-logic specifications: Progress and challenges," *AI Commun.*, vol. 29, no. 1, pp. 151–162, 2016.

[11] S. Söntges and M. Althoff, "Computing the drivable area of autonomous road vehicles in dynamic road scenes," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 6, pp. 1855–1866, 2018.

[12] S. Manzinger, C. Pek, and M. Althoff, "Using reachable sets for trajectory planning of automated vehicles," *IEEE Trans. Intell. Veh.*, vol. 6, no. 2, pp. 232–248, 2020.

[13] L. Schäfer, S. Manzinger, and M. Althoff, "Computation of solution spaces for optimization-based trajectory planning," *IEEE Trans. Intell. Veh.*, vol. 8, no. 1, pp. 216–231, 2021.

[14] G. Würsching and M. Althoff, "Sampling-based optimal trajectory generation for autonomous vehicles using reachable sets," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2021, pp. 828–835.

[15] H. Roehm, J. Oehlerking, T. Heinz, and M. Althoff, "STL model checking of continuous and hybrid systems," in *Int. Symp. Autom. Technol. Verif. Anal.*, 2016, pp. 412–427.

[16] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Tech., Model., Anal. Timed Fault-Tolerant Syst.*, 2004, pp. 152–166.

[17] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *Int. J. Rob. Res.*, vol. 20, no. 5, pp. 378–400, 2001.

[18] R. R. Da Silva, V. Kurtz, and H. Lin, "Automatic trajectory synthesis for real-time temporal logic," *IEEE Trans. Autom. Control*, vol. 67, no. 2, pp. 780–794, 2021.

[19] K. Esterle, V. Aravantinos, and A. Knoll, "From specifications to behavior: Maneuver verification in a semantic state space," in *Proc. of the IEEE Intell. Veh. Symp.*, 2019, pp. 2140–2147.

[20] C. K. Verginis, C. Vrohidis, C. P. Bechlioulis, K. J. Kyriakopoulos, and D. V. Dimarogonas, "Reconfigurable motion planning and control in obstacle cluttered environments under timed temporal tasks," in *Proc. of the IEEE Int. Conf. Robot. Autom.*, 2019, pp. 951–957.

[21] Y. Shoukry, P. Nuzzo, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada, "SMC: Satisfiability modulo convex programming," *Proc. of the IEEE*, vol. 106, no. 9, pp. 1655–1679, 2018.

[22] K. Cho, J. Suh, C. J. Tomlin, and S. Oh, "Cost-aware path planning under co-safe temporal logic specifications," *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 2308–2315, 2017.

[23] M. Lahijanian, M. R. Maly, D. Fried, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi, "Iterative temporal planning in uncertain environments with partial satisfaction guarantees," *IEEE Trans. Rob.*, vol. 32, no. 3, pp. 583–599, 2016.

[24] Y. Zhou, D. Maity, and J. S. Baras, "Timed automata approach for motion planning using metric interval temporal logic," in *Proc. of the Eur. Control Conf.*, 2016, pp. 690–695.

[25] D. Maity and J. S. Baras, "Motion planning in dynamic environments with bounded time temporal logic specifications," in *Mediterr. Conf. Control Autom.*, 2015, pp. 940–946.

[26] R. Kohlhaas, T. Bittner, T. Schamm, and J. M. Zöllner, "Semantic state space for high-level maneuver planning in structured traffic scenes," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2014, pp. 1060–1065.

[27] E. M. Wolff, U. Topcu, and R. M. Murray, "Automaton-guided controller synthesis for nonlinear systems with temporal logic," in *Proc. of the IEEE Int. Conf. Intell. Robot. Syst.*, 2013, pp. 4332–4339.

[28] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-time Syst.*, vol. 2, no. 4, pp. 255–299, 1990.

[29] A. Pnueli, "The temporal logic of programs," in *Annu. Symp. Found. of Comput. Sci.*, 1977, pp. 46–57.

[30] M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, *50 years of integer programming 1958-2008: From the early years to the state-of-the-art*. Springer Science & Business Media, 2009.

[31] D. Sun, J. Chen, S. Mitra, and C. Fan, "Multi-agent motion planning from signal temporal logic specifications," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 3451–3458, 2022.

[32] Z. Lin and J. S. Baras, "Optimization-based motion planning and runtime monitoring for robotic agent with space and time tolerances," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1874–1879, 2020.

[33] U. A. Fiaz and J. S. Baras, "Fast, composable rescue mission planning for UAVs using metric temporal logic," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 15 404–15 411, 2020.

[34] S. Saha and A. A. Julius, "An MILP approach for real-time optimal controller synthesis with metric temporal logic specifications," in *Proc. of the Am. Control Conf.*, 2016, pp. 1105–1110.

[35] E. M. Wolff and R. M. Murray, "Optimal control of nonlinear systems with temporal logic specifications," in *Rob. Res.*, 2016, pp. 21–37.

[36] C. I. Vasile, X. Li, and C. Belta, "Reactive sampling-based path planning with temporal logic specifications," *Int. J. Rob. Res.*, vol. 39, no. 8, pp. 1002–1028, 2020.

[37] J. Karlsson, F. S. Barbosa, and J. Tumova, "Sampling-based motion planning with temporal logic missions and spatial preferences," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 15 537–15 543, 2020.

[38] F. S. Barbosa, L. Lindemann, D. V. Dimarogonas, and J. Tumova, "Integrated motion planning and control under metric interval temporal logic specifications," in *Eur. Control Conf.*, 2019, pp. 2042–2049.

[39] C.-I. Vasile, V. Raman, and S. Karaman, "Sampling-based synthesis of maximally-satisfying controllers for temporal logic specifications," in *Proc. of the IEEE Int. Conf. Intell. Robot. Syst.*, 2017, pp. 3840–3847.

[40] L. I. R. Castro, P. Chaudhari, J. Tůmová, S. Karaman, E. Frazzoli, and D. Rus, "Incremental sampling-based algorithm for minimum-violation motion planning," in *Proc. of the IEEE Conf. Decis. Control*. IEEE, 2013, pp. 3217–3224.

[41] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Rob. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[42] L. Zhang and D. Manocha, "An efficient retraction-based RRT planner," in *Proc. of the IEEE Int. Conf. Robot. Autom.* IEEE, 2008, pp. 3743–3750.

[43] E. Irani Liu and M. Althoff, "Computing specification-compliant reachable sets for motion planning of automated vehicles," in *Proc. of the IEEE Intell. Veh. Symp.*, 2021, pp. 1037–1044.

[44] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *Proc. of the IEEE Intell. Veh. Symp.*, 2014, pp. 420–425.

[45] G. De Giacomo, A. Murano, F. Patrizi, and G. Perelli, "Timed trace alignment with metric temporal logic over finite traces," in *Proc. of the Int. Conf. Principles Knowl. Represent. and Reasoning*, vol. 18, no. 1, 2021, pp. 227–236.

[46] D. DSouza and P. Prabhakar, "On the expressiveness of MTL in the pointwise and continuous semantics," *Int. J. Software Tools for Technol. Transfer*, vol. 9, no. 1, pp. 1–4, 2007.

[47] A. Cecconi, C. D. Ciccio, G. D. Giacomo, and J. Mendling, "Interestingness of traces in declarative process mining: The Janus LTLp$_f$ approach," in *Int. Conf. Bus. Process Manage.* Springer, 2018, pp. 121–138.

[48] J. Ouaknine and J. Worrell, "Some recent results in metric temporal logic," in *Int. Conf. Formal Model. and Anal. Timed Syst.* Springer, 2008, pp. 1–13.

[49] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[50] A. Duret-Lutz, A. Lewkowicz, A. Fauchille, T. Michaud, E. Renault, and L. Xu, "Spot 2.0 – A framework for LTL and ω-automata manipulation," in *Int. Symp. Autom. Technol. Verif. Anal.*, 2016, pp. 122–129.

[51] G. Lafferriere, G. J. Pappas, and S. Yovine, "Symbolic reachability computation for families of linear vector fields," *J. Symb. Comput.*, vol. 32, no. 3, pp. 231–253, 2001.

[52] G. Holzmann, "Explicit-state model checking," in *Handbook of model checking*, 2018, pp. 153–170.

[53] O. Kupferman, "Automata theory and model checking," in *Handbook of model checking*, 2018, pp. 107–151.

[54] H. Tauriainen and K. Heljanko, "Testing LTL formula translation into Büchi automata," *Int. J. Software Tools for Technol. Transfer*, vol. 4, no. 1, pp. 57–70, 2002.

[55] P. Gastin and D. Oddoux, "Fast LTL to Büchi automata translation," in *Int. Conf. Comput. Aided Verif.* Springer, 2001, pp. 53–65.

[56] J. R. Büchi, "On a decision method in restricted second order arithmetic," in *Proc. of the Int. Congr. Logic, Method. and Philos. Sci.*, 1962, pp. 1–11.

[57] H. T. Croft, K. Falconer, and R. K. Guy, *Unsolved problems in geometry: Unsolved problems in intuitive mathematics*. Springer Science & Business Media, 2012, vol. 2.

[58] Y. Lin and M. Althoff, "Rule-compliant trajectory repairing using satisfiability modulo theories," in *Proc. of the IEEE Intell. Veh. Symp.*, 2022, pp. 449–456.

[59] C. Pek and M. Althoff, "Fail-safe motion planning for online verification of autonomous vehicles using convex optimization," *IEEE Trans. Rob.*, vol. 37, no. 3, pp. 798–814, 2020.

[60] B. Schürmann, D. Heß, J. Eilbrecht, O. Stursberg, F. Köster, and M. Althoff, "Ensuring drivability of planned motions using formal methods," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2017, pp. 1–8.

[61] M. Althoff and J. M. Dolan, "Reachability computation of low-order models for the safety verification of high-order road vehicle models," in *Proc. of the Am. Control Conf.*, 2012, pp. 3559–3566.

[62] J. Eilbrecht and O. Stursberg, "Challenges of trajectory planning with integrator models on curved roads," in *Proc. of the IFAC World Congr.*, 2020, pp. 15 588–15 595.

[63] C. Pek, V. Rusinov, S. Manzinger, M. C. Üste, and M. Althoff, "CommonRoad Drivability Checker: Simplifying the development and validation of motion planning algorithms," in *Proc. of the IEEE Intell. Veh. Symp.*, 2020, pp. 1013–1020.

[64] S. A. Kripke, "A completeness theorem in modal logic," *J. Symb. Log.*, vol. 24, no. 1, pp. 1–14, 1959.

[65] D. Gabbay, "The declarative past and imperative future," in *Temporal Log. Specification*, 1989, pp. 409–448.

[66] G. P. Maretić, M. T. Dashti, and D. Basin, "Anchored LTL separation," in *Proc. of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2014, pp. 1–9.

[67] I. M. Hodkinson and M. Reynolds, "Separation – past, present, and future." in *We Will Show Them! (2)*, 2005, pp. 117–142.

[68] N. Markey, "Temporal logic with past is exponentially more succinct," *Eur. Assoc. Theor. Comput. Sci.*, vol. 79, pp. 122–128, 2003.

[69] G. Roşu and K. Havelund, "Rewriting-based techniques for runtime verification," *Autom. Software Eng.*, vol. 12, no. 2, pp. 151–197, 2005.

[70] M. Reynolds, "More past glories," in *Proc. of the IEEE Symp. Logic. Comput. Sci.*, 2000, pp. 229–240.

[71] S. Dutta and M. Y. Vardi, "Assertion-based flow monitoring of SystemC models," in *Proc. of the ACM/IEEE Int. Conf. Formal Methods and Models Co-Des.*, 2014, pp. 145–154.

[72] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," in *Proc. of the Int. Joint Conf. Artif. Intell.* Association for Computing Machinery, 2013, pp. 854–860.

[73] A. Censi, K. Slutsky, T. Wongpiromsarn, D. Yershov, S. Pendleton, J. Fu, and E. Frazzoli, "Liability, ethics, and culture-aware behavior specification using rulebooks," in *Proc. of the IEEE Int. Conf. Robot. Autom.*, 2019, pp. 8536–8542.

[74] E. L. Lawler, *Combinatorial optimization: Networks and matroids*. Courier Corporation, 2001.

[75] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: composable benchmarks for motion planning on roads," in *Proc. of the IEEE Intell. Veh. Symp.*, 2017, pp. 719–726.

[76] E. Irani Liu, G. Würsching, M. Klischat, and M. Althoff, "CommonRoad-Reach: A toolbox for reachability analysis of automated vehicles," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2022, pp. 2313–2320.

[77] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a Frenet frame," in *Proc. of the IEEE Int. Conf. Robot. Autom.*, 2010, pp. 987–993.

[78] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *Am. J. Math.*, vol. 79, no. 3, pp. 497–516, 1957.

[79] Y. Lin and M. Althoff, "CommonRoad-CriMe: A toolbox for criticality measures of autonomous vehicles," in *Proc. of the IEEE Intell. Veh. Symp.*, 2023, pp. 1–8.

Edmond Irani Liu is currently a Ph.D. candidate and joined the Cyber-Physical Systems Group at the Technical University of Munich under Prof. Dr.-Ing. Matthias Althoff in 2019. He received his B.Sc. degree in automation in 2015 and his M.Sc. degree in control science and engineering in 2018, both from Shanghai Jiao Tong University, China. His research interests include specification-compliant reachability analysis and motion planning of automated vehicles.

Matthias Althoff received the diploma engineering degree in mechanical engineering and the Ph.D. degree in electrical engineering from the Technical University of Munich, Germany, in 2005 and 2010, respectively. He is currently an associate professor in computer science with the Technical University of Munich. From 2010 to 2012 he was a postdoctoral researcher with Carnegie Mellon University, Pittsburgh, USA, and from 2012 to 2013 an assistant professor at Ilmenau Technical University, Germany. His research interests include formal verification of continuous and hybrid systems, reachability analysis, planning algorithms, nonlinear control, automated vehicles, and power systems.

A.3 Specification-Compliant Motion Planning of Cooperative Vehicles Using Reachable Sets [3]

Summary Automated vehicles should explicitly adhere to enforced specifications to ensure their safe and effective engagement in mixed road traffic, in which both human-driven and automated vehicles co-exist. In addition to driving individually, infinitely many traffic scenarios exist in which inter-vehicle cooperation maximizes the collective benefits of the vehicles. This work addresses the problem of specification-compliant motion planning for a group of cooperating vehicles by leveraging reachability analysis techniques.

We provide a consolidated perspective on our previous works concerning the computation of specification-compliant reachable sets for a single vehicle [1], [2] and the negotiation of conflicting reachable sets within a group of cooperating vehicles [63]. The specification-compliant reachable sets of a vehicle represent the set of states that it can reach over time while complying with the considered specification. By systematically organizing the negotiation of the specification-compliant reachable sets of cooperating vehicles, each vehicle unambiguously receives its own negotiated reachable set, within which specification-compliant trajectories can be planned. Using an auction-based conflict resolution mechanism for reachable sets and forming a hierarchical structure of bids, we are able to reduce the exponential complexity inherent to trajectory-based collaborative motion planning of automated vehicles to only polynomial complexity. The effectiveness of our approach is evaluated using traffic scenarios from the CommonRoad benchmark suite.

Contributions of E. I. L. E. I. L. developed the idea of the research (together with M. A.); E. I. L. designed, conducted, and evaluated the experiments (together with M. A.); E. I. L. wrote the article (together with M. A.).

Book chapter The author's version of record of the article is reprinted. In press.

Specification-Compliant Motion Planning of Cooperative Vehicles Using Reachable Sets

Edmond Irani Liu and Matthias Althoff

Abstract Automated vehicles must comply explicitly with specifications, including traffic-based and handcrafted rules, in order for them to safely and effectively participate in mixed traffic. In addition to driving individually, there are many traffic situations in which cooperation between vehicles maximizes their collective benefits, including preventing collisions. To realize these benefits, we compute specification-compliant reachable sets for vehicles, i.e., sets of states which can be reached by vehicles over time that are constrained by a set of considered specifications. We summarize and combine our previous works on computing specification-compliant reachable sets and negotiating conflicting reachable sets within a group of cooperating vehicles. As a result, conflicts between specification-compliant reachable sets of vehicles are resolved, and specification-compliant trajectories can be individually planned for each vehicle within the negotiated reachable sets using arbitrary motion planners.

1 Introduction

When compared with human-driven vehicles, automated vehicles are expected to deliver enhanced road safety, passenger comfort, and traffic efficiency compared with human-driven vehicles. To safely and effectively participate in mixed traffic, in which both automated and human-driven vehicles share the road, automated vehicles must comply explicitly with specifications, including traffic regulations and handcrafted rules. Compliance with the former is essential in order to exempt manufacturers from liability claims in the event of an accident, while compliance with the latter allows motion plans to be generated that satisfy additional requirements. An example of a handcrafted rule is: *Follow vehicle 1 up to step k_1, then completely overtake it from the*

Edmond Irani Liu, Matthias Althoff
Technical University of Munich, Boltzmannstr. 3, 85748 Garching, Germany.
e-mail: {edmond.irani, althoff}@tum.de

left before step k_2. Generating a drivable trajectory that satisfies a set of specifications for an automated vehicle involves reasoning not only with continuous states (which may reflect the physical motion of the vehicle) but also discrete states (possibly due to discretization of the continuous state space or action space) of the vehicle. This poses computational challenges from a variety of aspects, including vehicle dynamics, the specifications under consideration (including collision avoidance), and dependencies between planned trajectories and constraints originating from the specifications. On the one hand, planning solely in the discrete state space may produce plans that meet specifications but violate vehicle dynamic constraints or lead to collisions. On the other hand, motion planners may generate dynamically drivable trajectories that do not comply with the specifications.

One solution to this problem is to guide the motion planning of an automated vehicle using its specification-compliant reachable set, which is defined as the set of states reachable by the vehicle over time that is constrained by a set of considered specifications. Computing the reachable sets in an over-approximative fashion will enclose all drivable trajectories of the automated vehicle [34]. The smaller the solution space is, the faster reachable sets can be computed, as demonstrated in [22]. In addition, the search space for the motion planner is greatly reduced particularly in critical situations. In contrast to conventional approaches, both effects result in quick computations even in critical situations. Low-level trajectory planning constraints can be extracted from the computed reachable sets and passed on to motion planners to generate specification-compliant trajectories.

In addition to driving individually, there are many traffic situations that demand cooperation between vehicles in order to maximize their collective benefits and to prevent collision in a potential emergency. Human drivers typically interact with each other through implicit communication and by anticipating the most likely behaviors of others. In comparison, automated vehicles can communicate and collaborate explicitly to jointly offer and suggest more sophisticated and efficient solutions in an ongoing traffic situation. One of the challenges of such cooperation lies in developing a computationally efficient scheme that does not compromise the optimality of the output solutions.

Reachable sets can be employed to tackle this challenge. The reachable sets of a group of cooperating vehicles can be computed and negotiated where conflicts in the position domain arise. This negotiation can be systematically organized such that each vehicle unambiguously receives its own negotiated reachable set, within which trajectories can be planned. This prevents exponential complexity of the collaborative motion planning.

In this chapter, we summarize and combine our previous works on computing specification-compliant reachable sets for an ego vehicle [13] as well as on negotiating conflicting reachable sets between a group of cooperating vehicles [21]. As a result, conflicts between specification-compliant reachable sets of vehicles are resolved, and each vehicle plans its own specification-compliant trajectories within its negotiated reachable set, for example, using the planners described in [36, 22].

The remainder of this article is organized as follows: Sect. 2 reviews related work on specification-compliant motion planning and cooperative motion planning.

Sect. 3 presents the necessary preliminaries and definitions. The computation of specification-compliant reachable sets is summarized in Sect. 4 and the negotiation of reachable sets in Sect. 5. Example results are presented in Sect. 6, and we conclude in Sect. 7.

2 Related Work

In this section, review related works on specification-compliant motion planning and cooperative motion planning of vehicles.

2.1 Specification-Compliant Motion Planning

The efforts to obtain a specification-compliant trajectory can be categorized on the basis of whether compliance with specifications is examined *after*, *during*, or *before* motion planning.

2.1.1 Considering Compliance After Motion Planning

The most straightforward approach to obtain a specification-compliant trajectory is to examine the compliance with specifications after the trajectories have been generated. The process of checking whether an execution of a system satisfies the expected behaviors is often referred to as *runtime verification* or *monitoring*. For example, article [29] presents a monitor for formally examining the compliance of automated vehicles with traffic rules (safe distances and overtaking); a monitor for so-called responsibility-sensitive safety rules [31] is described in [10]. While monitoring can be performed efficiently, monitors typically only provide a verdict, i.e., a *true* or *false* appraisal, on whether the specifications have been satisfied. If the trajectory under examination is rejected, no alternative trajectory is returned. This often necessitates the (re)planning of multiple trajectories in order to locate a valid solution for more complex specifications.

2.1.2 Considering Compliance During Motion Planning

Works in this category often adopt a mechanism that simultaneously handles planning in both the continuous and discrete state spaces of a system, with the generated discrete plans guiding the trajectory planning process. For example, a satisfiability modulo convex programming framework for cyber-physical systems was introduced in [32] that handles both convex constraints on a continuous model and Boolean constraints on a discrete model; article [16] puts forth a multilayered synergistic framework for motion planning of robots considering linear temporal logic (LTL); timed automata are used in [37] to synthesize timed paths for indoor robots that comply with specifications expressed in metric temporal logic. In these works, discrete plans are generated in the discrete state space based on abstractions of the considered systems, and trajectories are planned in the continuous state space by motion planners, with the discrete plans taken into consideration. In most cases, the dynamic constraints of the system are not reflected in the discrete plans. Thus, the drivability of these plans is often not ensured, requiring frequent replanning of both the discrete plans and the trajectories.

2.1.3 Considering Compliance Before Motion Planning

The final category of works considers the specifications prior to trajectory planning, e.g., in high-level maneuver planners, from which trajectory planning constraints can be extracted. The work in [15] generates maneuvers that respect simple traffic rules by traversing a graph defined in a discretized state space of the ego vehicle; article [8] embraces a similar concept and produces maneuvers satisfying specifications expressed in LTL; in [33], so-called driving corridors are extracted from reachable sets of an ego vehicle that reflect different position relations to other vehicles over time. Our approach to computing specification-compliant reachable sets [13] falls into this category. It can handle propositional logic with predicates related to positions, velocities, accelerations, and certain traffic regulations introduced in [18, 19].

2.2 Cooperative Motion Planning

Survey articles [9, 24, 28] reviewed recent advances in cooperative driving of automated vehicles with varied focuses on architecture, maneuver planning, and motion planning use cases. Optimization-based and reservation-based approaches are common paradigms for cooperative motion planning [9, 28]. In optimization-based approaches, one or more optimization problems are formulated based on the motion planning constraints and cost functions of cooperating vehicles. The optimization problems are solved with a (centralized) optimizer, which corresponds to trajectories to be followed by the cooperating vehicles. The complexity of the optimization problem increases dramatically with the number of vehicles considered, which requires either a high computation power or a limit to the number of vehicles in a group.

Our approach to cooperative motion planning falls into the reservation-based category and employs auction algorithms for resolving conflicts in reachable sets of vehicles. Reservation-based methods assign free space to vehicles for trajectory planning. Earlier works with a focus on intersection management were introduced by Dresner [4]: Tiles are created from the intersection region, which can be requested by vehicles approaching the intersection. A centralized intersection manager proceeds to assign tiles with multiple requests to vehicles, using a first-come-first-served protocol, ensuring that no tile is occupied by more than one vehicle at any one time. Its extensions and variations are presented in [6, 5]. As the first-come-first-served policy for reservation assignment may be inefficient in situations with higher traffic density, it was replaced in [27, 3, 35] by auction-based methods. In auction-based methods, each bidder (cooperating vehicle) bids for offered packages (e.g., combinations of tiles representing road areas) in a way that reflects its interests or utilities. An auction algorithm is then executed to maximize the total revenue of the packages. Instead of tiles, some works identify possible conflicting points, regions, or moving space-time corridors and allocate them to vehicles in the event of a conflict [20, 17, 38, 23]. The corridors correspond to predefined behaviors, such as following a lane or performing a lane change; vehicles receiving such corridors must act accordingly. In [25, 11], an efficient and explicit space-time reservation protocol was devised for cooperative maneuver planning, through which a vehicle broadcasts requested space envelopes over time and drives within the envelopes once the request has been accepted by surrounding vehicles of interest.

3 Preliminaries

This section introduces the necessary preliminaries, including the general setup, coordinate systems, definitions of reachable sets, and propositional logic.

3.1 Setup and Coordinate System

In this work, the considered scenarios are described in the CommonRoad[1] [1] format, which consists of (1) a road network constructed of lanelets [2], whose left and right bounds are represented by polylines, (2) dynamic and static obstacles, and (3) traffic rule elements (such as road markings, traffic signs, and traffic lights). Fig. 1 depicts an exemplary traffic scenario. We denote by $\mathcal{V}^{\mathrm{c}} = \{V_1^{\mathrm{c}}, \dots, V_N^{\mathrm{c}}\}$ the set of cooperative vehicles V_n^{c} with IDs $\mathcal{N} = \{1, \dots, N\}$ for which trajectories are planned. Each V_n^{c} is associated with a planning problem with a planning horizon of up to $k_h \in \mathbb{N}_0$, which includes the initial state of V_n^{c} and a set of goal states. A reference path Γ_n is constructed for a planning problem with a given route planner, which is then used to establish a local curvilinear coordinate system F_n^{L} of V_n^{c} as described in [2]. Within F_n^{L}, (s_n, d_n) describes the longitudinal coordinate s_n and the lateral coordinate d_n. Adopting this coordinate system facilitates the formulation of maneuvers from the perspective of V_n^{c}, examples of which include lane-following and preventing driving backwards. We use $\mathcal{LL}$ to denote the set of lanelets in the road network of a considered scenario. Without loss of generality, we assume obstacles present in the scenarios to be non-cooperating vehicles, denoted by $\mathcal{V}^{\mathrm{o}} = \{V_1^{\mathrm{o}}, \dots, V_M^{\mathrm{o}}\}$

[1] https://commonroad.in.tum.de/

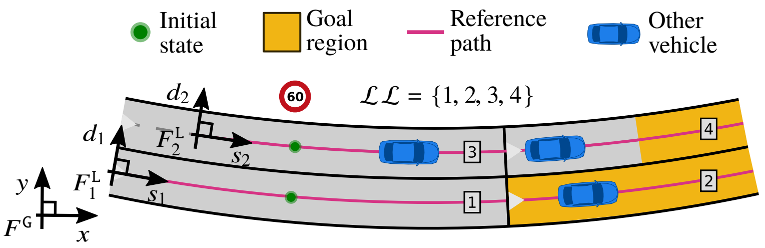


Fig. 1: A scenario containing planning problems with two cooperating ego vehicles V_1^{c} and V_2^{c}, and four lanelets with IDs 1–4. The triangles at the beginning of each lanelet indicate the driving directions

with IDs $\mathcal{M} = \{1, \dots, M\}$. In addition, we assume that the most likely predictions of trajectories of other vehicles V_m^{o} are given as input. The conflicts between the reachable sets of vehicles in $\mathcal{V}^{\mathrm{c}}$ are detected and resolved in the global Cartesian coordinate system F^{G}.

3.2 System Dynamics

The dynamics of an ego vehicle V_n^{c} is abstracted by a point-mass model with the center of the vehicle as the reference point. Notably, the reachable sets of the point-mass model over-approximate those of high-fidelity vehicle models; thus, this abstraction does not exclude possible behaviors of V_n^{c}. This model is represented with two double integrators in its longitudinal s_n and lateral d_n directions. Let $\square_n$ be a variable of V_n^{c}, with minimum and maximum values denoted by $\underline{\square}_n$ and $\overline{\square}_n$, respectively. The system dynamics of V_n^{c} is

$$\boldsymbol{x}_{n,k+1} = f(\boldsymbol{x}_{n,k}, \boldsymbol{u}_{n,k}) = \begin{pmatrix} 1 & \Delta_t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta_t \\ 0 & 0 & 0 & 1 \end{pmatrix} \boldsymbol{x}_{n,k} + \begin{pmatrix} \frac{1}{2}\Delta_t^2 & 0 \\ \Delta_t & 0 \\ 0 & \frac{1}{2}\Delta_t^2 \\ 0 & \Delta_t \end{pmatrix} \boldsymbol{u}_{n,k}, \tag{1}$$

where $k \in \mathbb{N}_0$ is a step corresponding to time $t_k = k\Delta_t$, with $\Delta_t \in \mathbb{R}_+$ being a predefined time increment. The variable $\boldsymbol{x}_{n,k} \in \mathcal{X}_{n,k} \subset \mathbb{R}^4$ represents the state of V_n^{c} in the state space $\mathcal{X}_{n,k}$, and $\boldsymbol{u}_{n,k} \in \mathcal{U}_{n,k} \subset \mathbb{R}^2$ represents an input in the input space $\mathcal{U}_{n,k}$ of V_n^{c}, each at step k. The states and inputs are modeled as $\boldsymbol{x}_{n,k} = (s_{n,k}, \dot{s}_{n,k}, d_{n,k}, \dot{d}_{n,k})^{\mathsf{T}}$ and $\boldsymbol{u}_{n,k} = (\ddot{s}_{n,k}, \ddot{d}_{n,k})^{\mathsf{T}}$, respectively. The velocities and accelerations at a position $(s_{n,k}, d_{n,k})$ are bounded by

$$\underline{\dot{s}}(\Gamma_n) \leq \dot{s}_{n,k} \leq \overline{\dot{s}}(\Gamma_n), \;\; \underline{\dot{d}}(\Gamma_n) \leq \dot{d}_{n,k} \leq \overline{\dot{d}}(\Gamma_n), \tag{2a}$$

$$\underline{\ddot{s}}(\Gamma_n) \leq \ddot{s}_{n,k} \leq \overline{\ddot{s}}(\Gamma_n), \;\; \underline{\ddot{d}}(\Gamma_n) \leq \ddot{d}_{n,k} \leq \overline{\ddot{d}}(\Gamma_n). \tag{2b}$$

The bounds are chosen conservatively to consider the kinematic limitations and effects of representing the system dynamics using the point-mass model within a curvilinear coordinate system, see, for example, article [7]. We define an operator $\mathrm{proj}_{\diamond}(\cdot)$ for subsequent computations, which maps the input to its elements $\diamond$. An example is: $\mathrm{proj}_{(s,\dot{s})}(\tilde{\boldsymbol{x}}_{n,k}) = (s_{n,k}, \dot{s}_{n,k})^{\mathsf{T}}$ for $\tilde{\boldsymbol{x}}_{n,k} = (s_{n,k}, \dot{s}_{n,k}, \ddot{s}_{n,k})^{\mathsf{T}}$. A set $\tilde{\mathcal{X}}_{n,k}$ can be projected using the same operator:

$$\mathrm{proj}_{\diamond}(\tilde{\mathcal{X}}_{n,k}) = \left\{ \mathrm{proj}_{\diamond}(\tilde{\boldsymbol{x}}_{n,k}) \middle| \tilde{\boldsymbol{x}}_{n,k} \in \tilde{\mathcal{X}}_{n,k} \right\}.$$

3.3 Reachable Set

We denote the occupancy of V_n^{c} by $Q_n(\boldsymbol{x}_{n,k}) \subset \mathbb{R}^2$ and the occupancies of all vehicles in $\mathcal{V}^{\mathrm{o}}$ as well as the regions outside the road surface by $O_{n,k} \subset \mathbb{R}^2$, both within F_n^{L}. The set of forbidden states $\mathcal{X}_{n,k}^{\mathrm{F}}$ of V_n^{c} at k is defined as

$$\mathcal{X}_{n,k}^{\mathrm{F}} := \left\{ \boldsymbol{x}_{n,k} \in \mathcal{X}_{n,k} \middle| Q_n(\boldsymbol{x}_{n,k}) \cap O_{n,k} \neq \emptyset \right\}.$$

Let $\mathcal{R}_{n,0}^{*} = \mathcal{X}_{n,0}$ be the initial reachable set of V_n^{c}, with $\mathcal{X}_{n,0}$ being the initial set of states. The reachable set $\mathcal{R}_{n,k+1}^{*}$ of the next step is defined as the set of states reachable from the current reachable set $\mathcal{R}_{n,k}^{*}$ while avoiding the forbidden states:

$$\begin{aligned}\mathcal{R}_{n,k+1}^{*} := \Big\{ \boldsymbol{x}_{n,k+1} \in \mathcal{X}_{n,k+1} \Big| & \exists \boldsymbol{x}_{n,k} \in \mathcal{R}_{n,k}^{*}, \exists \boldsymbol{u}_{n,k} \in \mathcal{U}_{n,k} : \\ & \boldsymbol{x}_{n,k+1} = f(\boldsymbol{x}_{n,k}, \boldsymbol{u}_{n,k}) \wedge \boldsymbol{x}_{n,k+1} \notin \mathcal{X}_{n,k+1}^{\mathrm{F}} \Big\}.\end{aligned}$$

Efficient computation of $\mathcal{R}_{n,k}^{*}$ is generally difficult; hence, we compute its over-approximation $\mathcal{R}_{n,k} \approx \mathcal{R}_{n,k}^{*}$, which encloses all trajectories of V_n^{c}. We adopt the union of so-called *base sets* $\mathcal{R}_{n,k}^{(i)}$, $i \in \mathbb{N}$ as a set representation for $\mathcal{R}_{n,k}$ [34]. Each base set $\mathcal{R}_{n,k}^{(i)} = \hat{\mathcal{P}}_{s,n,k}^{(i)} \times \hat{\mathcal{P}}_{d,n,k}^{(i)}$ is chosen to be a Cartesian product of two convex polytopes that enclose the reachable positions and velocities of V_n^{c} in the $(s_n, \dot{s}_n)$ and $(d_n, \dot{d}_n)$ planes, respectively (see Fig. 2a–b). To simplify the notation, we also denote the collection (set of sets) of $\mathcal{R}_{n,k}^{(i)}$ by $\mathtt{R}_{n,k} = \left\{ \mathcal{R}_{n,k}^{(1)}, \ldots, \mathcal{R}_{n,k}^{(i)}, \ldots \right\}$. The projection of $\mathcal{R}_{n,k}^{(i)}$ onto the position domain yields axis-aligned rectangles $\mathcal{D}_{n,k}^{(i)}$ (see Fig. 2c), whose union is referred to as the drivable area $\mathcal{D}_{n,k}$. Similarly, we use $\mathtt{D}_{n,k}$ to denote the collection of $\mathcal{D}_{n,k}^{(i)}$.

In this study, each base set $\mathcal{R}_{n,k}^{(i)}$ carries a set of semantic labels $\mathcal{L}_{n,k}^{(i)}$, whose collection is denoted by $\mathtt{L}_{n,k}$. The generation of $\mathcal{L}_{n,k}^{(i)}$ will be explained in Sect. 4.6.3. To store the relationships of $\mathcal{R}_{n,k}^{(i)}$ in terms of reachability and time, we create a directed and acyclic graph G_n, which is referred to as a *reachability graph*, see Fig. 3.

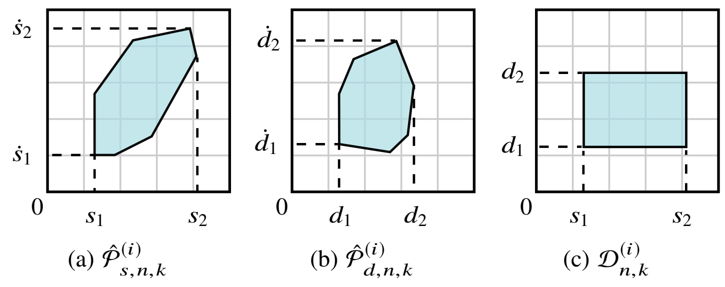
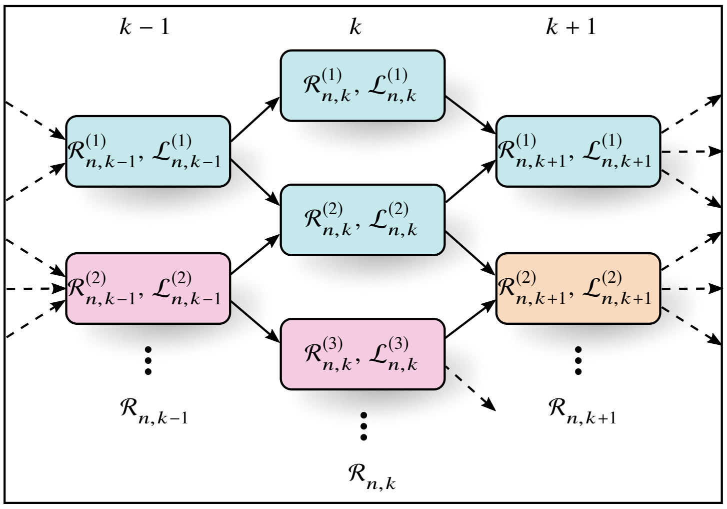


Fig. 2: Polytopes and drivable area of a base set $\mathcal{R}^{(i)}_{n,k}$ (adapted from [13])



$k-1$
k
$k+1$
$\mathcal{R}^{(1)}_{n,k-1}, \mathcal{L}^{(1)}_{n,k-1}$
$\mathcal{R}^{(2)}_{n,k-1}, \mathcal{L}^{(2)}_{n,k-1}$
$\mathcal{R}_{n,k-1}$
$\mathcal{R}^{(1)}_{n,k}, \mathcal{L}^{(1)}_{n,k}$
$\mathcal{R}^{(2)}_{n,k}, \mathcal{L}^{(2)}_{n,k}$
$\mathcal{R}^{(3)}_{n,k}, \mathcal{L}^{(3)}_{n,k}$
$\mathcal{R}_{n,k}$
$\mathcal{R}^{(1)}_{n,k+1}, \mathcal{L}^{(1)}_{n,k+1}$
$\mathcal{R}^{(2)}_{n,k+1}, \mathcal{L}^{(2)}_{n,k+1}$
$\mathcal{R}_{n,k+1}$

Fig. 3: Reachability graph G_n connecting nodes of different steps. Nodes of the same color have the same labels (adapted from [13])

Each node in G_n corresponds to one base set with its labels. An edge connecting $\mathcal{R}^{(i)}_{n,k}$ and $\mathcal{R}^{(j)}_{n,k+1}$ indicates that $\mathcal{R}^{(j)}_{n,k+1}$ is reachable from $\mathcal{R}^{(i)}_{n,k}$ after one step.

3.4 Propositional Logic

We consider specifications expressed in propositional logic [12] for V^{c}_n, denoted by $\mathcal{F}_n$, which are directly integrated during the computation of the reachable sets (see Sect. 4.6.4). Let $\varphi_n \in \mathcal{F}_n$ be a propositional logic formula, we introduce an additional syntax $\mathbf{G}_I(\varphi_n)$, $I = [a, b]$, $0 \le a \le b \le k_h$, where I is an integer interval specifying steps for which φ_n should hold. If I is not specified, we assume it to be the entire planning horizon $[0, k_h]$. For example, the following specification requires V^{c}_n to follow V^{o}_1 between steps 0 and 10, and never to be on the right of V^{o}_1:

Table 1: Selection of considered predicates inspired by [19] (adapted from [13])

Category	Type	Predicate
Position	VI	in_lanelet, on_main_carriageway, on_access_ramp, ...
	VD	behind, beside, in_front_of, left_of, aligned_with, right_of, ...
Velocity	VI	below_fov_velocity_limit, below_type_velocity_limit, ...
	VD	safe_following_velocity_speed_limit, safe_leading_velocity_speed_limit, ...
Acceleration	VD	admissible_braking, ...
General	VI	change_lanelet, preserve_traffic_flow, standing_still, ...
	VD	in_congestion, exists_slow_leading_vehicle, ...

$$\mathbf{G}_{[0,10]}\left(\text{behind}(V_1^{\mathrm{o}}) \wedge \text{aligned\_with}(V_1^{\mathrm{o}})\right) \wedge \mathbf{G}\left(\neg\, \text{right\_of}(V_1^{\mathrm{o}})\right).$$

4 Computing Specification-Compliant Reachable Sets

To obtain specification-compliant and negotiated reachable sets for V_n^{c}, we (1) semantically label reachable sets considering relevant predicates, (2) constrain reachable sets to subsets satisfying specifications $\mathcal{F}_n$, and (3) negotiate conflicting reachable sets with other cooperating vehicles in $\mathcal{V}^{\mathrm{c}}$. This section summarizes our previous work [13] covering steps 1 and 2; step 3 will be covered in the next section. A selection of considered predicates is listed in Tab. 1: The evaluation of a *vehicle-dependent (VD)* predicate is dependent on other vehicles $\mathcal{V}^{\mathrm{o}}$, whereas that of a *vehicle-independent (VI)* predicate is not.

4.1 State Space Partitioning

To expedite the labeling of reachable sets, we partition the state space of V_n^{c} based on considered position predicates. Velocity predicates are not considered in the partitioning since they require computationally demanding splitting of the state space of V_n^{c} with (non)linear curves (see Fig. 5c-d). For efficiency, we instead directly evaluate them on individual reachable sets (see Sect. 4.6.2). Set operations such as intersection and difference are required to compute the partitions of the state space. We model the partitions for V_n^{c} with a set of hyperrectangles $R_{n,q}$ to avoid gross approximations while keeping computational complexity at a reasonable level. This choice is not mandatory; any other set representation that captures the partitions will also suffice. $R_{n,q}$ is defined as the Cartesian product of intervals over the position and velocity domains within F_n^{L}:

$$R_{n,q} := \left([\underline{s}_{n,q}, \overline{s}_{n,q}] \times [\underline{\dot{s}}_{n,q}, \overline{\dot{s}}_{n,q}]\right) \times \left([\underline{d}_{n,q}, \overline{d}_{n,q}] \times [\underline{\dot{d}}_{n,q}, \overline{\dot{d}}_{n,q}]\right), \tag{3}$$

where $s_{n,q}$ and $\dot{s}_{n,q}$ denote the position and velocity of the q-th hyperrectangle in the s_n direction, respectively. The same applies to $d_{n,q}$ and $\dot{d}_{n,q}$ in the d_n direction. A regular grid of axis-aligned cells is formed along Γ_n and the q-th cell in the grid occupies $[\underline{s}_{n,q}, \overline{s}_{n,q}] \times [\underline{d}_{n,q}, \overline{d}_{n,q}] \subset \mathbb{R}^2$. The default values of the velocity intervals $[\underline{\dot{s}}_{n,q}, \overline{\dot{s}}_{n,q}]$ and $[\underline{\dot{d}}_{n,q}, \overline{\dot{d}}_{n,q}]$ are set according to (2a).

The set of considered position predicates as well as its power set are denoted by $\mathcal{P}^{\mathrm{pos}} = \{\sigma_1, \sigma_2, \dots\}$ and $2^{\mathcal{P}^{\mathrm{pos}}}$, respectively. We also denote by $\mathrm{part}_n(k; \mathcal{Z}_{n,j})$ the set of hyperrectangles of $V_n^{\mathtt{c}}$ for which the predicates in $\mathcal{Z}_{n,j} \in \mathcal{Z}_n \subseteq 2^{\mathcal{P}^{\mathrm{pos}}}$ evaluate to true at step k. Fig. 4 and Fig. 5b illustrate example partitions projected onto the (s_n, d_n) and $(s_n, \dot{s}_n)$ planes, respectively.

4.2 Position Predicates

For a concise presentation, we present only a few example evaluations position predicates. Vehicle-independent position predicates do not depend on other vehicles; examples are:

- in_lanelet($R_{n,q}; L_{\mathtt{id}}$) $\Leftrightarrow \mathrm{proj}_{(s,d)}(R_{n,q}) \cap \mathrm{occ}_n(L_{\mathtt{id}}) \neq \emptyset$, where $L_{\mathtt{id}} \in \mathcal{LL}$ denotes the lanelet with ID $\mathtt{id}$, and $\mathrm{occ}_n(L_{\mathtt{id}})$ returns its occupancy within $F_n^{\mathtt{L}}$.
- drives_rightmost($R_{n,q}; \mathcal{X}^{\mathtt{RM}}$) $\Leftrightarrow \mathrm{proj}_{(s,d)}(R_{n,q}) \cap \mathcal{X}^{\mathtt{RM}} \neq \emptyset$, where $\mathcal{X}^{\mathtt{RM}} \subset \mathbb{R}^2$ denotes the rightmost region of lanelets. Within this region, the distance between any point to the right bound of a lanelet does not exceed a predefined distance [19].

For the sake of brevity, we omit $R_{n,q}$ in the arguments of the predicates in the rest of this work.

Vehicle-dependent position predicates describe position relationships between an ego vehicle $V_n^{\mathtt{c}}$ and non-cooperating vehicles in $\mathcal{V}^{\mathtt{o}}$. Following [19], we define necessary helper functions to assist the evaluation of predicates. The functions front($k; n; m$) and rear($k; n; m$) return the maximum and minimum longitudinal coordinates of $V_m^{\mathtt{o}}$ within $F_n^{\mathtt{L}}$, respectively, each at step k. Along the longitudinal direction, the mutually exclusive predicates $\mathcal{P}^{\mathtt{pos}}_{n,m,s} = \big\{\{\text{in\_front\_of}(V_m^{\mathtt{o}})\}, \{\text{behind}(V_m^{\mathtt{o}})\}, \{\text{beside}(V_m^{\mathtt{o}})\}\big\}$ can be evaluated as follows:

- in_front_of($V_m^{\mathtt{o}}$) $\Leftrightarrow \underline{s}_{n,q} - l_n/2 > \mathrm{front}(k; n; m)$,
- behind($V_m^{\mathtt{o}}$) $\Leftrightarrow \overline{s}_{n,q} + l_n/2 < \mathrm{rear}(k; n; m)$,
- beside($V_m^{\mathtt{o}}$) $\Leftrightarrow \neg$ in_front_of($V_m^{\mathtt{o}}$) $\wedge \neg$ behind($V_m^{\mathtt{o}}$) $\wedge$ (left_of($V_m^{\mathtt{o}}$) $\vee$ right_of($V_m^{\mathtt{o}}$)).

We define the mutually exclusive set of predicates $\mathcal{P}^{\mathtt{pos}}_{n,m,d} = \big\{\{\text{left\_of}(V_m^{\mathtt{o}})\}, \{\text{right\_of}(V_m^{\mathtt{o}})\}, \{\text{aligned\_with}(V_m^{\mathtt{o}})\}\big\}$ similarly along the lateral direction.

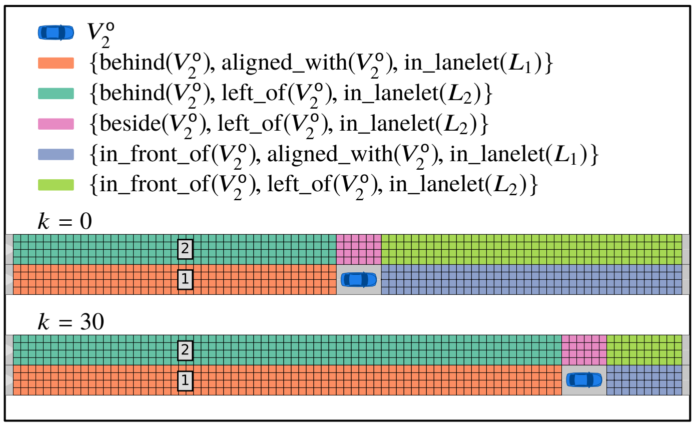

V_2^{o}
{behind(V_2^{o}), aligned_with(V_2^{o}), in_lanelet(L_1)}
{behind(V_2^{o}), left_of(V_2^{o}), in_lanelet(L_2)}
{beside(V_2^{o}), left_of(V_2^{o}), in_lanelet(L_2)}
{in_front_of(V_2^{o}), aligned_with(V_2^{o}), in_lanelet(L_1)}
{in_front_of(V_2^{o}), left_of(V_2^{o}), in_lanelet(L_2)}
$k = 0$
2
1
$k = 30$
2
1

Fig. 4: Projection of the partitions of realizable sets of position predicates onto the position domain. Lanelet IDs are shown with numbered boxes. In this example we only consider position predicates related to L_1, L_2, and V_2^{o} (adapted from [13])

4.3 Realizable Sets of Position Predicates

The partitions of the collection $\mathcal{Z}_n$ of realizable sets of position predicates of V_n^{c} are used for splitting the reachable sets (see Sect. 4.6.2). Sets $\mathcal{Z}_{n,j} \in \mathcal{Z}_n$ are said to be realizable for V_n^{c} if $\exists k \in 0, \ldots, k_h : \mathrm{part}_n(k; \mathcal{Z}_{n,j}) \neq \emptyset$, with k_h being the planning horizon. We refer the readers to [13, Sec. III.C] for the computation of $\mathcal{Z}_n$. Fig. 4 shows an example of the partitions of $\mathcal{Z}_{n,j}$ projected onto the position domain for a scenario containing two lanelets and one non-cooperating vehicle. It follows from our formulation of the predicates that the aforementioned projection is collision-free with respect to other vehicles.

4.4 Velocity Predicates

We briefly present examples of the evaluation of velocity predicates required for the subsequent computation of reachable sets. Vehicle-independent velocity predicates often relate to extremum requirements on velocities. For example, rule R-G3 [19] specifies maximum velocity limits originating from different sources, which should be respected. These include velocity limits introduced by the type of lane(let), the type of vehicle, and the limited field of view of the ego vehicle.

The evaluation of vehicle-dependent velocity predicates depends on other vehicles $\mathcal{V}^{\mathrm{o}}$. Examples are predicates indicating whether the ego vehicle V_n^{c} is driving at a safe velocity with respect to a leading or a following vehicle V_m^{o} [19, cf. Sec. IV.C]. See [19] for further examples.

Algorithm 1 One-Step Computation of Specification-Compliant Reachable Sets

```
Inputs: Specifications $\mathcal{F}_n$, base sets $\mathcal{R}_{n,k-1}$, realizable sets of predicates $\mathcal{Z}_n$.
Output: Updated reachability graph $G_n$.
1: $\mathcal{R}^{\mathrm{P}}_{n,k} \leftarrow$ Propagate($\mathcal{R}_{n,k-1}$)    ▷ Sect. 4.6.1
2: $\mathcal{R}^{\mathrm{S}}_{n,k} \leftarrow$ Split($\mathcal{R}^{\mathrm{P}}_{n,k}$,$\mathcal{Z}_n$)    ▷ Sect. 4.6.2
3: $\mathcal{L}_{n,k} \leftarrow$ Label($\mathcal{R}^{\mathrm{S}}_{n,k}$,$\mathcal{F}_n$)    ▷ Sect. 4.6.3
4: CheckCompliance($\mathcal{R}^{\mathrm{S}}_{n,k}$,$\mathcal{L}_{n,k}$,$\mathcal{F}_n$)    ▷ Sect. 4.6.4
5: $\mathcal{R}_{n,k} \leftarrow$ CreateNewBaseSets($\mathcal{R}^{\mathrm{S}}_{n,k}$)    ▷ Sect. 4.6.5
6: for $\mathcal{R}^{(i)}_{n,k} \in \mathcal{R}_{n,k}$ do
7:     $G_n$.AddNode($\mathcal{R}^{(i)}_{n,k}$,$\mathcal{L}^{(i)}_{n,k}$)
8: end for
```

4.5 General Traffic Situation Predicates

General traffic situation predicates may reveal the states of a cooperating or non-cooperating vehicle. These include whether V^{c}_n or V^{o}_m has conducted a lane change maneuver, whether a slow leading vehicle exists for V^{c}_n, and whether V^{c}_n is stuck in traffic congestion. See [19] for further examples.

4.6 Computation of Reachable Sets

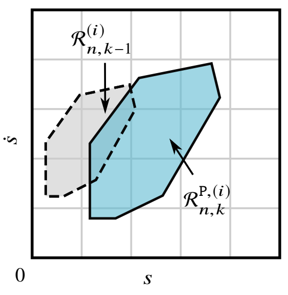
Alg. 1 details one step of the computation of specification-compliant reachable sets for an ego vehicle. The reachable sets of subsequent steps are computed analogously.

4.6.1 Forward Propagation

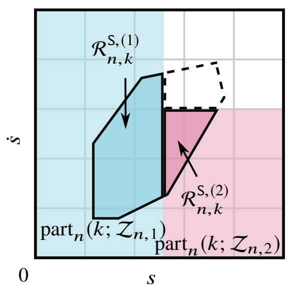
Each base set $\mathcal{R}^{(i)}_{n,k-1} \in \mathcal{R}_{n,k-1}$ from the previous step is forward-propagated based on the discrete-time system model (1), resulting in the propagated sets $\mathcal{R}^{\mathrm{P},(i)}_{n,k}$ (see Fig. 5a). We perform the forward propagation as described in [34], except that additional acceleration constraints originating from the specifications can be imposed (for example, unnecessary braking rule R_G2 in [19]).

4.6.2 Splitting

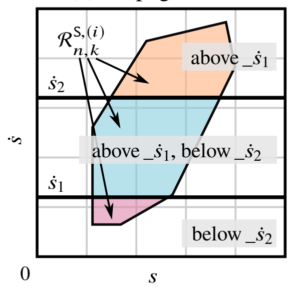
The propagated sets $\mathcal{R}^{\mathrm{P},(i)}_{n,k}$are split into new sets $\mathcal{R}^{\mathrm{S},(i)}_{n,k}$ with respect to position and velocity predicates:


$\mathcal{R}^{(i)}_{n,k-1}$
$\mathcal{R}^{\mathrm{P},(i)}_{n,k}$
$\dot{s}$
0
s

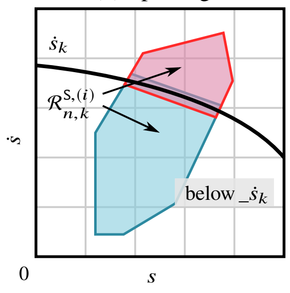
(a) Propagation


$\mathcal{R}^{\mathrm{S},(1)}_{n,k}$
$\mathcal{R}^{\mathrm{S},(2)}_{n,k}$
$\mathrm{part}_n(k;\mathcal{Z}_{n,1})$
$\mathrm{part}_n(k;\mathcal{Z}_{n,2})$
$\dot{s}$
0
s

(b) Splitting


$\mathcal{R}^{\mathrm{S},(i)}_{n,k}$
above_$\dot{s}_1$
$\dot{s}_2$
above_$\dot{s}_1$, below_$\dot{s}_2$
$\dot{s}_1$
below_$\dot{s}_2$
$\dot{s}$
0
s

(c) Splitting with respect to vehicle-independent velocity predicates above_$\dot{s}_1$ and below_$\dot{s}_2$



(d) Splitting with respect to vehicle-dependent velocity predicate below_$\dot{s}_k$

Fig. 5: Propagation, splitting, and labeling of base sets. We only show the operations in the s-direction. Labels of polytopes are shown in gray boxes. Notably, in (d), the two newly split polytopes are slightly over-approximated and convexified due to the nonlinearity introduced by the velocity predicate (adapted from [13])

1. $\mathcal{R}^{\mathrm{P},(i)}_{n,k}$ are split such that the new sets only intersect with a single partition (see Fig. 5b).
2. The split sets are further split, over-approximated, and convexified with respect to velocity predicates (see Fig. 5c–d).

4.6.3 Semantic Labeling

The semantic labels $\mathcal{L}^{(i)}_{n,k}$ of reachable sets $\mathcal{R}^{\mathrm{S},(i)}_{n,k}$ are updated as follows:

1. $\mathcal{R}^{\mathrm{S},(i)}_{n,k}$ propagated with acceleration-specific specifications include atomic propositions $\sigma \in \mathcal{AP}$ corresponding to acceleration predicates in their set of labels.
2. $\mathcal{R}^{\mathrm{S},(i)}_{n,k}$ include atomic propositions $\sigma \in \mathcal{AP}$ corresponding to the position predicates associated with the partition with which it intersects, velocity predicates, and traffic situation predicates that hold in $\mathcal{R}^{\mathrm{S},(i)}_{n,k}$ in their set of labels.

4.6.4 Compliance Check

In this step, we iterate through $\mathcal{R}^{\mathrm{S},(i)}_{n,k}$ and examine the compliance of the labels $\mathcal{L}^{(i)}_{n,k}$ with the given specifications $\mathcal{F}_n$. We discard $\mathcal{R}^{\mathrm{S},(i)}_{n,k}$ if $\exists \varphi_n \in \mathcal{F}_n : \mathcal{L}^{(i)}_{n,k} \not\models \varphi_n$. If all sets are discarded, $\mathcal{F}_n$ cannot be complied with by any trajectory of the ego vehicle (recall that our reachable sets are over-approximative). In this case, one can either recompute the reachable sets with respect to a different set of specifications or execute a previously computed fail-safe trajectory [26].

4.6.5 Creation of New Base Sets

Finally, the new base sets are created by computing the drivable areas $\mathcal{D}^{\mathrm{S},(i)}_{n,k}$ of $\mathcal{R}^{\mathrm{S},(i)}_{n,k}$, repartitioning $\mathcal{D}^{\mathrm{S},(i)}_{n,k}$, and producing $\mathcal{R}^{(i)}_{n,k}$. We refer the reader to [34] for a detailed explanation of these steps. The reachability graph G_n is updated by adding $\mathcal{R}^{(i)}_{n,k}$ along $\mathcal{L}^{(i)}_{n,k}$ as new nodes.

5 Negotiation of Reachable Sets

This section summarizes our previous work on the negotiation of conflicting reachable sets $\mathcal{R}_{n,k}$ among a group of cooperating vehicles [21]. We use the notation $[\square_n]_1^N = [\square_1, \dots, \square_N]$ to denote a list of elements $\square_n$ of vehicles V_n^{c}. Alg. 2 details the steps for resolving conflicts between cooperating vehicles at each step k:

1. Compute specification-compliant reachable sets for each cooperating vehicle.
2. Identify conflicting cells based on reachable sets of cooperating vehicles (see Sect. 5.1).
3. Determine the optimal allocation of packages of cells among cooperating vehicles (see Sect. 5.2).
4. Compute negotiated reachable sets for each cooperating vehicle (see Sect. 5.2).

Step 1 is computed as described in Sect. 4; we will now elaborate on steps 2–4.

5.1 Problem Statement

We denote by $C = \{C_0, C_1, \dots, C_{\tilde{i}}, \dots\}$ a grid with cells $C_{\tilde{i}}$ of rectangular shape, created by tessellation of the position domain within the global Cartesian coordinate system F^{G}. Each cell is an individual asset representing an area of the road surface and can be combined into unions of assets, which we refer to as packages $C_{\tilde{j}}$. We specify the mapping $\mathrm{cell}_n : 2^{\mathcal{X}_{n,k}} \to 2^{C}$ that returns the cells $C_{\tilde{i}} \in C$ occupied by vehicle V_n^{c} due to its set of states $\mathcal{X}_{n,k}$ at step k and its shape. The cooperating vehicles

Algorithm 2 Computation of Negotiated Reachable Sets

```
1: function ComputeNegotiatedReachableSet($[\mathcal{R}_{n,0}]_1^N$, $C$)
2:     $[\mathcal{R}^{\mathrm{N}}_{n,0}]_1^N \leftarrow [\mathcal{R}_{n,0}]_1^N$                ▷ Initialization
3:     for $k = 1$ to $k_h$ do
4:         for $n = 1$ to $N$ do
5:             $\mathcal{R}_{n,k} \leftarrow$ ComputeReachableSet($\mathcal{R}^{\mathrm{N}}_{n,k-1}$)          ▷ Sect. 4.6
6:         end for
7:         $C_k^{\mathrm{C}} \leftarrow$ IdentifyConflictingCells($[\mathcal{R}_{n,k}]_1^N$, $C$)          ▷ Sect. 5.1
8:         $\mathcal{W}^* \leftarrow$ DetermineOptimalAllocation($[\mathcal{R}_{n,k}]_1^N$, $C_k^{\mathrm{C}}$)      ▷ Sect. 5.2
9:         for $n = 1$ to $N$ do
10:            $\mathcal{R}^{\mathrm{N}}_{n,k} \leftarrow$ ComputeNegotiatedReachableSet($\mathcal{R}_{n,k}$, $\mathcal{W}^*$)   ▷ Sect. 5.2
11:        end for
12:    end for
13:    return $[\cup_k \mathcal{R}^{\mathrm{N}}_{n,k}]_1^N$
14: end function
```

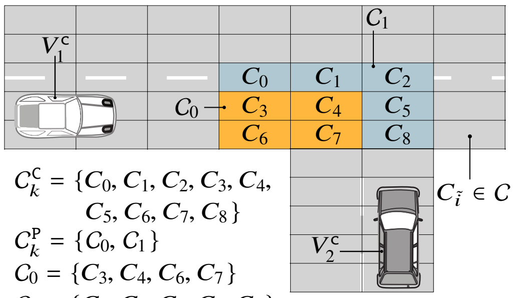
in $\mathcal{V}^{\mathrm{c}}$ act as bidders and propose bids to packages $C_{\tilde{j}}$ for which $C_{\tilde{j}} \cap \mathrm{cell}_n(\mathcal{R}_{n,k}) \neq \emptyset$ holds. Let us introduce $2^{\mathcal{N}}_{>=2}$ to denote all subsets of the power set of $\mathcal{N}$ with a cardinality greater than one, $C_k^{\mathrm{C}} \subseteq C$ denotes the set of conflicting cells requested by at least two vehicles at step k:

$$C_k^{\mathrm{C}} := \bigcup_{\mathcal{I} \in 2^{\mathcal{N}}_{>=2}} \bigcap_{n \in \mathcal{I}} \mathrm{cell}_n(\mathcal{R}_{n,k}). \tag{4}$$

We restrict the packages to those containing at least one conflicting cell, denoted by $C_k^{\mathrm{P}} \subseteq 2^{C_k^{\mathrm{C}}}$ (see Fig. 6). We assume that every cooperating vehicle V_n^{c} bids its true value, with $\overline{b}_k(C_{\tilde{j}})$ being the maximum bid of the package $C_{\tilde{j}}$ proposed by $\mathcal{V}^{\mathrm{c}}$. The overall revenue is maximized, while no single cell is assigned to multiple bidders:

$$\max_{\delta_k(C_{\tilde{j}})} \sum_{C_{\tilde{j}}} \delta_k(C_{\tilde{j}})\, \overline{b}_k(C_{\tilde{j}}), \tag{5}$$

where $\delta_k(C_{\tilde{j}}) = 1$ if package $C_{\tilde{j}}$ is assigned to the bidder with the highest bid at step k. Problem (5) is known as the *winner determination problem*, and its solution is NP-hard [30]. Furthermore, accepting every package $C_{\tilde{j}}$ demands that each bidder V_n^{c} bids for $2^{|C_k^{\mathrm{C}}|} - 1$ packages at step k, which becomes more computationally demanding as $|C_k^{\mathrm{C}}|$ grows. Using a hierarchical tree structure for the packages allows us to attain computational tractability and ensures that the optimal allocation of packages will be found in the time $O(|C_k^{\mathrm{C}}|^2)$ [30].



$$\mathcal{C}_k^{\mathrm{C}} = \{C_0, C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8\}$$
$$\mathcal{C}_k^{\mathrm{P}} = \{\mathcal{C}_0, \mathcal{C}_1\}$$
$$\mathcal{C}_0 = \{C_3, C_4, C_6, C_7\}$$
$$\mathcal{C}_1 = \{C_0, C_1, C_2, C_5, C_8\}$$

Fig. 6 Visualization of the road grid $\mathcal{C}$, the set of conflicting cells $\mathcal{C}_k^{\mathrm{C}}$, the set of packages $\mathcal{C}_k^{\mathrm{P}}$, and the individual packages $\mathcal{C}_{\tilde{j}}$ (adapted from [21])

5.2 Conflict Resolution

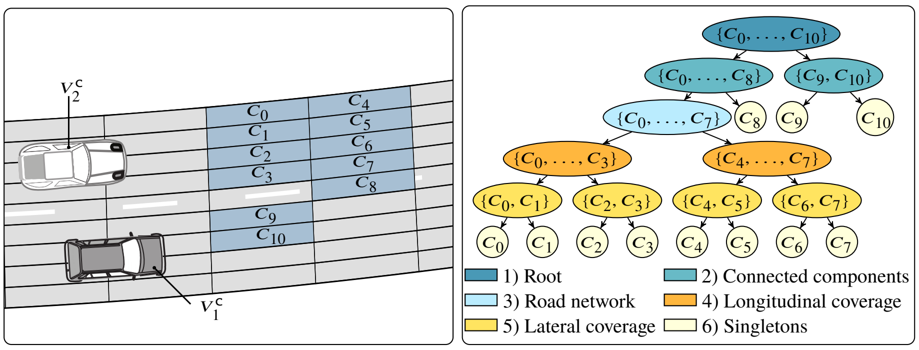
We employ an auction-based mechanism to resolve conflicts with occupied road cells between cooperating vehicles. At every step k, the conflicts are resolved as follows:

1. Determine packages $\mathcal{C}_{\tilde{j}}$ based on $\mathcal{C}_k^{\mathrm{C}}$ and their position within the hierarchical tree (see Sect. 5.2.1).
2. Evaluate individual bids of packages $\mathcal{C}_{\tilde{j}}$ and determine the maximum bid $\overline{b}_k(\mathcal{C}_{\tilde{j}})$ (see Sect. 5.2.2).
3. Determine the optimal allocation $\mathcal{W}^*$ of packages to cooperating vehicles (see Sect. 5.2.3).

5.2.1 Hierarchical Tree of Packages

All conflicting cells $\mathcal{C}_k^{\mathrm{C}}$ at k are included in the root node of a hierarchical tree T. At each level of the tree, the cells in a parent node are decomposed into disjoint sets of cells, each of which is a package associated with a child node (see Fig. 7). To decompose the cells into more granular packages, we consider the following levels:

1. Connected components: Connected regions on the road surface prevents ego vehicles having disjointed drivable areas, which would complicate subsequent motion planning. We aggregate connected cells into packages.
2. Road network: Vehicles have to obey the traffic rules imposed by the road network; therefore, we encourage the creation of packages based on lanelets. A cell is assigned to the lanelet with which it has the largest intersecting area.
3. Longitudinal position coverage: The packages of the parent nodes are decomposed in the longitudinal direction such that the longitudinal coverage of each new package does not exceed a predefined threshold.
4. Lateral position coverage: The packages of the parent nodes are decomposed in the lateral direction such that the lateral coverage of each new package does not exceed a predefined threshold.
5. Singletons: The packages comprise only a single cell.



(a) V_1^c intends to perform a lane change maneuver; both V_1^c and V_2^c request cells $\{C_0,\dots,C_{10}\}$

(b) A possible hierarchical tree constructed using the decomposition strategy outlined in Sect. 5.2.1

Fig. 7: Example grouping of conflicting cells (adapted from [21])

5.2.2 Bids on Packages

We adopt a common utility function for cooperating vehicles to avoid a situation in which a vehicle could continuously outbid others due to differences in the scales and weights used to calculate the bids on packages. We use the following sets as the basis for computing the utility of V_n^c for $C_{\tilde{j}}$ to determine $b_{n,k}(C_{\tilde{j}})$:

1. the conflict-free reachable set: $\mathcal{R}_{n,k}^{\mathsf{CF}} := \left\{x_{n,k} \in \mathcal{R}_{n,k} \middle| \mathrm{cell}_n(\{x_{n,k}\}) \cap C_k^{\mathsf{C}} = \emptyset\right\}$.
2. the conflicting reachable set depending on package $C_{\tilde{j}}$ that would be lost if $C_{\tilde{j}}$ was not assigned to V_n^c: $\mathcal{R}_{n,k}^{\mathsf{CP}}(C_{\tilde{j}}) := \left\{x_{n,k} \in \mathcal{R}_{n,k} \middle| \mathrm{cell}_n(\{x_{n,k}\}) \cap C_{\tilde{j}} \neq \emptyset\right\}$.
3. the assigned reachable set that V_n^c possesses given that $C_{\tilde{j}}$ is assigned to V_n^c: $\mathcal{R}_{n,k}^{\mathsf{AS}}(C_{\tilde{j}}) := \mathcal{R}_{n,k}^{\mathsf{CF}} \cup \mathcal{R}_{n,k}^{\mathsf{CP}}(C_{\tilde{j}})$.

For computational reasons, the sets $\mathcal{R}_{n,k}^{\mathsf{CF}}$,$\mathcal{R}_{n,k}^{\mathsf{CP}}(C_{\tilde{j}})$, and $\mathcal{R}_{n,k}^{\mathsf{AS}}(C_{\tilde{j}})$ are approximated by the union of base sets (see Sect. 3.3) and are denoted by $\cup_i \mathcal{R}_{n,k}^{\mathsf{CF},(i)}$, $\cup_i \mathcal{R}_{n,k}^{\mathsf{CP},(i)}$, and $\cup_i \mathcal{R}_{n,k}^{\mathsf{AS},(i)}$, respectively. To take the objectives of the vehicles into account while preventing the complete loss of the reachable set of a vehicle (so that a trajectory can still be found), the utilities of vehicles are computed differently for regular mode and survival mode:

$$b_{n,k}(C_{\tilde{j}}) := \begin{cases} U_{n,k}^{\mathsf{R}}(C_{\tilde{j}}), & \mathrm{area}(\mathcal{R}_{n,k}^{\mathsf{CF}}) > \underline{A}, \text{(regular mode)} \\ U_{n,k}^{\mathsf{S}}(C_{\tilde{j}}), & \text{otherwise, (survival mode)} \end{cases}$$

where area$(\cdot)$ returns the size of the drivable area of the input (see Sect. 3.3) and $\underline{A}$ is a threshold. We now proceed with explaining the regular mode and survival mode.

1) *Regular Mode:* The utility of $\mathcal{R}_{n,k}^{\mathsf{AS}}$ (or $\mathcal{R}_{n,k}^{\mathsf{CF}}$, as the case may be) is defined as the sum of the utilities of $\mathcal{R}_{n,k}^{\mathsf{AS},(i)}$ (or $\mathcal{R}_{n,k}^{\mathsf{CF},(i)}$), weighted by their areas. The function

$U^{\mathtt{R}}_{n,k}(C_{\tilde{j}})$ reflects the utility of $C_{\tilde{j}}$ for $V^{\mathtt{c}}_n$ by computing the ratio of the utility of $\mathcal{R}^{\mathtt{AS}}_{n,k}$ to that of $\mathcal{R}^{\mathtt{CF}}_{n,k}$:

$$U^{\mathtt{R}}_{n,k}(C_{\tilde{j}}) = \frac{\sum_i \left(u^{\mathtt{pos}}(\mathcal{R}^{\mathtt{AS},(i)}_{n,k}) + u^{\mathtt{ref}}(\mathcal{R}^{\mathtt{AS},(i)}_{n,k})\right) \times \text{area}\left(\mathcal{R}^{\mathtt{AS},(i)}_{n,k}\right)}{\sum_i \left(u^{\mathtt{pos}}(\mathcal{R}^{\mathtt{CF},(i)}_{n,k}) + u^{\mathtt{ref}}(\mathcal{R}^{\mathtt{CF},(i)}_{n,k})\right) \times \text{area}\left(\mathcal{R}^{\mathtt{CF},(i)}_{n,k}\right)},$$

with partial utility functions $u^{\mathtt{pos}}$ and $u^{\mathtt{ref}}$. To encourage advances in traffic flow, we reward progression in the longitudinal direction with

$$u^{\mathtt{pos}}(\square_{n,k}) = y\left(\frac{\max(\text{proj}_{(s)}(\square_{n,k})) - \max(\text{proj}_{(s)}(\mathcal{R}^{\mathtt{N}}_{n,k-1}))}{\frac{1}{2}\,\overline{\ddot{s}}_{n,k}\,\Delta_t^2 + \overline{\dot{s}}_{n,k}\,\Delta_t}\right),$$

where $\overline{\ddot{s}}_{n,k}$ and $\overline{\dot{s}}_{n,k}$ are determined according to (2a), and y is a generalized logistic function that maps the utility to $(0, 1)$; in addition to [21], we also consider the deviation of $V^{\mathtt{c}}_n$ from its reference path:

$$u^{\mathtt{ref}}(\square_{n,k}) = e^{-w\,d'},\, d' = \min(\{|d''|\,|d'' \in \text{proj}_{(d)}(\square_{n,k})\}),$$

where $w \in \mathbb{R}_+$ is a tunable weight that dictates how fast $u^{\mathtt{ref}}(\square_{n,k})$ approaches 0 as the deviation increases.

2) *Survival Mode:* Two countermeasures are introduced to prevent reachable sets of $V^{\mathtt{c}}_n$ from vanishing: (1) if any $V^{\mathtt{c}}_n$ is in survival mode, no other vehicle in regular mode can bid on the package $C_{\tilde{j}}$; (2) the utility function is switched to

$$U^{\mathtt{S}}_{n,k}(C_{\tilde{j}}) = \frac{\text{area}\left(\mathcal{R}^{\mathtt{CP}}_{n,k}(C_{\tilde{j}})\right)}{\text{area}\left(\mathcal{R}_{n,k}\right)},$$

which reflects how close the reachable set of $V^{\mathtt{c}}_n$ is to vanishing given that $C_{\tilde{j}}$ is not assigned to $V^{\mathtt{c}}_n$.

5.2.3 Optimal Allocation of Packages

The algorithm for finding the optimal allocation $\mathcal{W}^*$ of packages $C_{\tilde{j}}$ is based on [30]. In each iteration, we retrieve the deepest node $N^{\mathtt{deep}}$ in the hierarchical tree T (see Sect. 5.2.1), its parent node $N^{\mathtt{parent}}$, and the set of child nodes $\mathcal{N}^{\mathtt{child}} = \left\{\dots, N^{\mathtt{child}}_{\tilde{q}}, \dots\right\}$ of $N^{\mathtt{parent}}$. Next, we compare the summed maximum bids (revenue) of all child nodes $\text{rev}(\mathcal{N}^{\mathtt{child}}) := \sum_{\tilde{q}} \overline{b}_k(N^{\mathtt{child}}_{\tilde{q}})$ with the maximum bid of the parent node $\overline{b}_k(N^{\mathtt{parent}})$:

- If $\overline{b}_k(N^{\mathtt{parent}}) > \text{rev}(\mathcal{N}^{\mathtt{child}})$, $\mathcal{N}^{\mathtt{child}}$ is excluded from $\mathcal{W}^*$.
- If $\overline{b}_k(N^{\mathtt{parent}}) \leq \text{rev}(\mathcal{N}^{\mathtt{child}})$, $N^{\mathtt{parent}}$ is excluded from $\mathcal{W}^*$.

Following this comparison, $\mathcal{N}^{\mathtt{child}}$ is removed from the tree. The process is repeated until $\mathcal{N}^{\mathtt{parent}}$ becomes the root node. After obtaining $\mathcal{W}^*$, each ego vehicle $V_n^{\mathtt{c}}$ proceeds to determine its negotiated reachable sets:

$$\mathcal{R}_{n,k}^{\mathtt{N}} := \left\{ x_{n,k} \in \mathcal{R}_{n,k} \middle| \operatorname{cell}_n(\{x_{n,k}\}) \cap C_{n,k}^{\mathtt{UA}} = \emptyset \right\},$$

where $C_{n,k}^{\mathtt{UA}} \subseteq C_k^{\mathtt{C}}$ denotes the set of unassigned cells of $V_n^{\mathtt{c}}$ based on $\mathcal{W}^*$.

6 Evaluation

This section provides example results for specification-compliant reachable sets for a single ego vehicle and its extension to cooperative vehicles. The implementation is based on the CommonRoad-Reach toolbox [14] for computing the reachable sets of vehicles.

6.1 Scenario I: Precise Overtaking

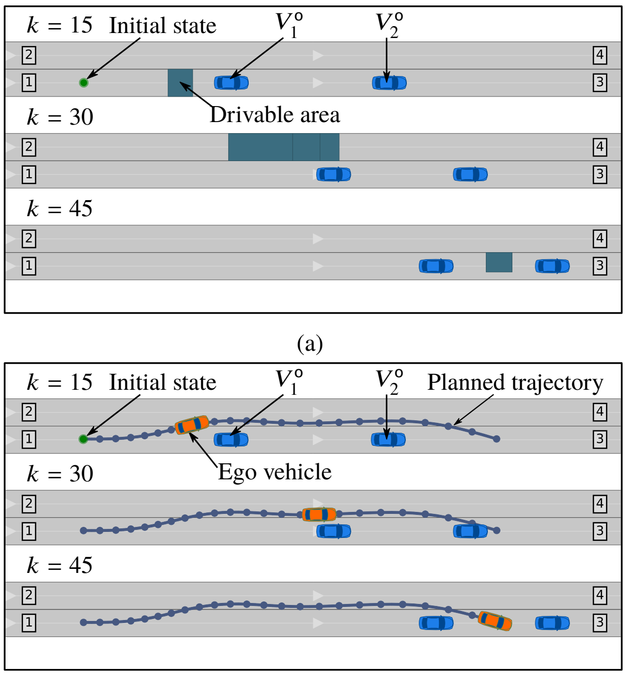
This scenario depicts a situation in which the vehicle $V_1^{\mathtt{c}}$ should overtake a leading vehicle $V_1^{\mathtt{o}}$ in the presence of another vehicle $V_2^{\mathtt{o}}$. Let the following specification be issued by a high-level maneuver planner of $V_1^{\mathtt{c}}$:

$$\begin{aligned}
&\mathbf{G}_{[0,15]}\left(\text{behind}(V_1^{\mathtt{o}}) \wedge \text{aligned\_with}(V_1^{\mathtt{o}})\right) \wedge \\
&\mathbf{G}_{[16,38]}\left(\text{in\_lanelet}(L_2) \vee \text{in\_lanelet}(L_4)\right) \wedge \\
&\mathbf{G}_{[39,45]}\left(\text{in\_front\_of}(V_1^{\mathtt{o}}) \wedge \text{behind}(V_2^{\mathtt{o}}) \wedge \text{in\_lanelet}(L_3)\right).
\end{aligned}$$

The specification-compliant reachable sets are computed as described in Sect. 4. The non-empty result implies that it is possible to find a trajectory that meets the specifications. Fig. 8 visualizes the drivable areas of $V_1^{\mathtt{c}}$ along with a trajectory planned within the reachable sets using the motion planner described in [22]. For a more detailed evaluation of computing specification-compliant reachable sets for a single ego vehicle, we refer the reader to [13].

6.2 Scenario II: Highway

In this scenario, we negotiate the reachable sets of four cooperating vehicles driving on a highway. Fig. 9 shows the computation results at different steps. As can be seen, our method can allocate road areas to cooperating vehicles even in such complex traffic situations with many non-cooperating traffic participants. For a more detailed



(a)

(b)

Fig. 8: Overtaking scenario. (a) Drivable area at different steps. (b) A trajectory planned within the reachable set (adapted from [13])

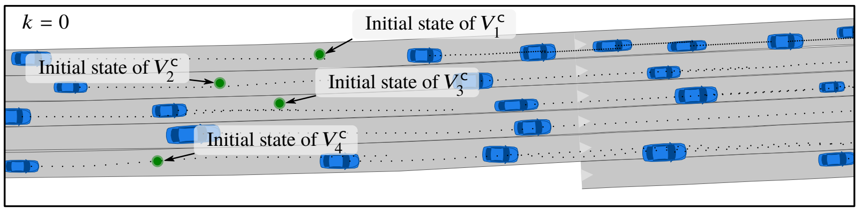
evaluation of negotiating reachable sets among a group of cooperating vehicles, we refer the reader to [21].

6.3 Scenario III: Roundabout

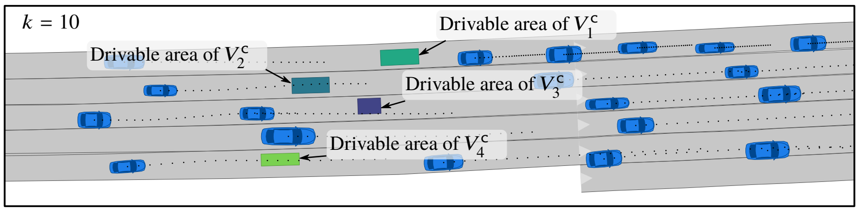
This scenario illustrates a situation in which two vehicles V_1^c and V_2^c should cooperate to go around a roundabout. We show the computation results under different settings: (1) no specification is considered; (2) V_1^c yields to V_2^c; (3) V_2^c yields to V_1^c. The latter two settings are relevant when a *yield* traffic sign is present at the junction and specifies which vehicle has to yield to other vehicles entering with a higher passing priority. The specification can be expressed as follows:

$$\mathbf{G}(\text{exists\_yield\_sign} \land \text{exists\_other\_entering\_vehicle} \Rightarrow \text{brake\_to\_stop}),$$

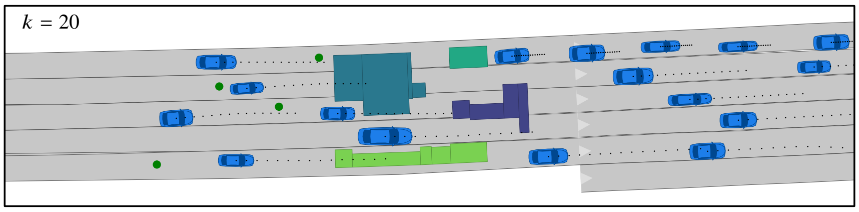
which can be regarded as a simplified version of the intersection rules described in [18] but without temporal logic connectives. Fig. 10 illustrates the computation results under these settings. In Fig. 10b, V_2^c can either accelerate and enter the



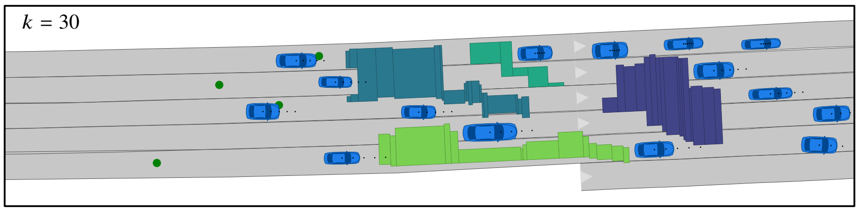
(a)



(b)



(c)



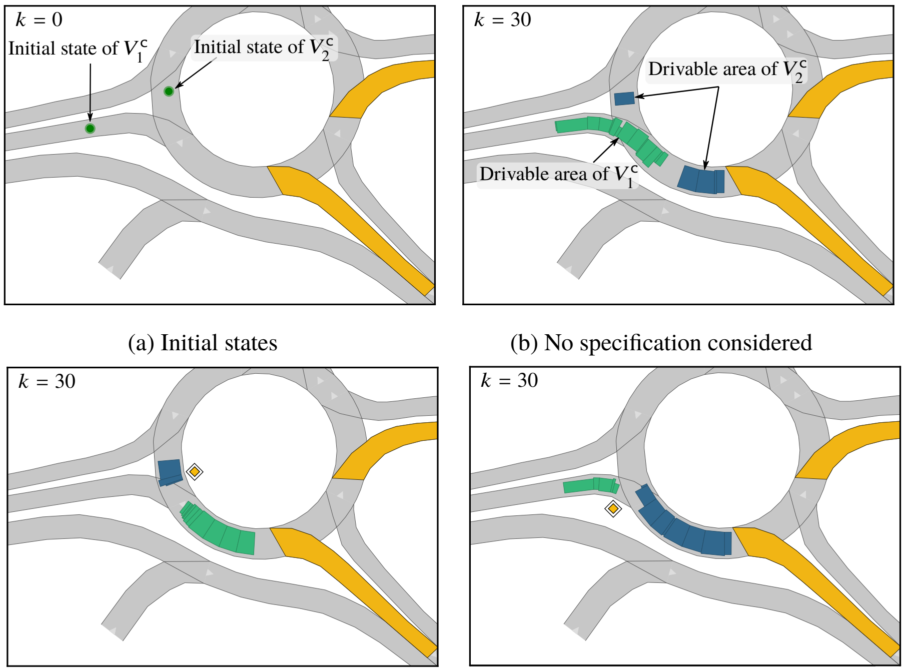
(d)

Fig. 9: Highway scenario. Subfigures b–d show the drivable areas of the negotiated reachable sets of vehicles at different steps

roundabout ahead of V_1^c or decelerate to enable V_1^c to enter first. In Fig. 10c–d, the yielding vehicles have to brake in order to stop and yield to the other entering vehicle.

7 Conclusions

In this chapter, we summarized our previous works on computing specification-compliant reachable sets for an ego vehicle and negotiating conflicting reachable sets between a group of cooperating vehicles. The specification-compliant and negotiated reachable set is used to guide subsequent motion planners to find specification-



(a) Initial states (b) No specification considered

(c) Vehicle V_2^c yields (d) Vehicle V_1^c yields

Fig. 10: Roundabout scenario. V_2^c intends to reach the first exit, and V_1^c intends to reach the second exit. Subfigures b–d show the drivable areas of the negotiated reachable sets of vehicles at step $k = 30$

compliant trajectories. As a result, the cooperative vehicles can consider traffic rules and handcrafted rules expressed in propositional logic that involve position, velocity, acceleration, and general traffic situation predicates. A limitation of the method is that it does not yet handle specifications formulated in temporal logic, which reflects temporal requirements on vehicles, both in the computation and negotiation of the reachable sets. This will be a subject of future research.

Acknowledgements This work was funded by the Deutsche Forschungsgemeinschaft (German Research Foundation) within the Priority Program SPP 1835 *Cooperative Interacting Automobiles* under grant No. AL 1185/4-2, Huawei-TUM collaboration project *Research on Key Technologies of Safety Assurance for Autonomous Vehicles*, and the Central Innovation Program of the German Federal Government under grant No. KK5116401KG0. The authors appreciate the fruitful collaborations with the project partners.

References

1. Althoff, M., Koschi, M., Manzinger, S.: CommonRoad: composable benchmarks for motion planning on roads. In: Proc. of the IEEE Intell. Veh. Symp., pp. 719–726 (2017)
2. Bender, P., Ziegler, J., Stiller, C.: Lanelets: Efficient map representation for autonomous driving. In: Proc. of the IEEE Intell. Veh. Symp., pp. 420–425 (2014)
3. Carlino, D., Boyles, S.D., Stone, P.: Auction-based autonomous intersection management. In: Proc. of the IEEE Int. Conf. Intell. Transp. Syst., pp. 529–534 (2013)
4. Dresner, K., Stone, P.: Multiagent traffic management: A reservation-based intersection control mechanism. In: Proc. of the Int. Joint Conf. Auton. Agents and Multiagent Syst., pp. 530–537 (2004)
5. Dresner, K., Stone, P.: Turning the corner: improved intersection control for autonomous vehicles. In: Proc. of the IEEE Intell. Veh. Symp., pp. 423–428 (2005)
6. Dresner, K., Stone, P.: Human-usable and emergency vehicle-aware control policies for autonomous intersection management. In: Workshop on Agents in Traffic and Transp., pp. 17–25 (2006)
7. Eilbrecht, J., Stursberg, O.: Challenges of trajectory planning with integrator models on curved roads. In: Proc. of the IFAC World Congr., pp. 15588–15595 (2020)
8. Esterle, K., Aravantinos, V., Knoll, A.: From specifications to behavior: maneuver verification in a semantic state space. In: Proc. of the IEEE Intell. Veh. Symp., pp. 2140–2147 (2019)
9. Häfner, B., Bajpai, V., Ott, J., Schmitt, G.A.: A survey on cooperative architectures and maneuvers for connected and automated vehicles. IEEE Commun. Surv. Tutorials **24**(1), 380–403 (2021)
10. Hekmatnejad, M., Yaghoubi, S., Dokhanchi, A., Amor, H.B., Shrivastava, A., Karam, L., Fainekos, G.: Encoding and monitoring responsibility sensitive safety rules for automated vehicles in signal temporal logic. In: Proc. of the ACM/IEEE Int. Conf. Formal Method. Model. Syst. Des., pp. 1–11 (2019)
11. Heß, D., Lattarulo, R., Pérez, J., Schindler, J., Hesse, T., Köster, F.: Fast maneuver planning for cooperative automated vehicles. In: Proc. of the IEEE Int. Conf. Intell. Transp. Syst., pp. 1625–1632 (2018)
12. Huth, M., Ryan, M.: Logic in computer science: Modelling and reasoning about systems. Cambridge university press (2004)
13. Irani Liu, E., Althoff, M.: Computing specification-compliant reachable sets for motion planning of automated vehicles. In: Proc. of the IEEE Intell. Veh. Symp., pp. 1037–1044 (2021)
14. Irani Liu, E., Würsching, G., Moritz, K., Althoff, M.: CommonRoad-Reach: A toolbox for reachability analysis of automated vehicles. In: Proc. of the IEEE Int. Conf. Intell. Transp. Syst., pp. 1–8 (2022)
15. Kohlhaas, R., Bittner, T., Schamm, T., Zöllner, J.M.: Semantic state space for high-level maneuver planning in structured traffic scenes. In: Proc. of the IEEE Int. Conf. Intell. Transp. Syst., pp. 1060–1065 (2014)
16. Lahijanian, M., Maly, M.R., Fried, D., Kavraki, L.E., Kress-Gazit, H., Vardi, M.Y.: Iterative temporal planning in uncertain environments with partial satisfaction guarantees. IEEE Trans. Rob. **32**(3), 583–599 (2016)
17. Levin, M.W., Fritz, H., Boyles, S.D.: On optimizing reservation-based intersection controls. IEEE Trans. Intell. Transp. Syst. **18**(3), 505–515 (2017)
18. Maierhofer, S., Moosbrugger, P., Althoff, M.: Formalization of intersection traffic rules in temporal logic. In: Proc. of the IEEE Intell. Veh. Symp., pp. 1135–1144 (2022)
19. Maierhofer, S., Rettinger, A.K., Mayer, E.C., Althoff, M.: Formalization of interstate traffic rules in temporal logic. In: Proc. of the IEEE Intell. Veh. Symp., pp. 752–759 (2020)
20. Manzinger, S., Althoff, M.: Negotiation of drivable areas of cooperative vehicles for conflict resolution. In: Proc. of the IEEE Int. Conf. Intell. Transp. Syst., pp. 1–8 (2018)
21. Manzinger, S., Althoff, M.: Tactical decision making for cooperative vehicles using reachable sets. In: Proc. of the IEEE Int. Conf. Intell. Transp. Syst., pp. 444–451 (2018)

22. Manzinger, S., Pek, C., Althoff, M.: Using reachable sets for trajectory planning of automated vehicles. IEEE Trans. Intell. Veh. **6**(2), 232–248 (2020)
23. Marinescu, D., Čurn, J., Bouroche, M., Cahill, V.: On-ramp traffic merging using cooperative intelligent vehicles: A slot-based approach. In: Proc. of the IEEE Int. Conf. Intell. Transp. Syst., pp. 900–906 (2012)
24. Montanaro, U., Dixit, S., Fallah, S., Dianati, M., Stevens, A., Oxtoby, D., Mouzakitis, A.: Towards connected autonomous driving: review of use-cases. Veh. Syst. Dyn. **57**(6), 779–814 (2019)
25. Nichting, M., Heß, D., Schindler, J., Hesse, T., Köster, F.: Space time reservation procedure (STRP) for V2X-based maneuver coordination of cooperative automated vehicles in diverse conflict scenarios. In: Proc. of the IEEE Intell. Veh. Symp., pp. 502–509 (2020)
26. Pek, C., Althoff, M.: Fail-safe motion planning for online verification of autonomous vehicles using convex optimization. IEEE Trans. Rob. **37**(3), 798–814 (2020)
27. Rewald, H., Stursberg, O.: Cooperation of autonomous vehicles using a hierarchy of auction-based and model-predictive control. In: Proc. of the IEEE Intell. Veh. Symp., pp. 1078–1084 (2016)
28. Rios-Torres, J., Malikopoulos, A.A.: A survey on the coordination of connected and automated vehicles at intersections and merging at highway on-ramps. IEEE Trans. Intell. Transp. Syst. **18**(5), 1066–1077 (2017)
29. Rizaldi, A., Keinholz, J., Huber, M., Feldle, J., Immler, F., Althoff, M., Hilgendorf, E., Nipkow, T.: Formalising and monitoring traffic rules for autonomous vehicles in Isabelle/HOL. In: Int. Conf. Integr. Formal Methods, pp. 50–66 (2017)
30. Rothkopf, M.H., Pekeč, A., Harstad, R.M.: Computationally manageable combinational auctions. Manage. Sci. **44**(8), 1131–1147 (1998)
31. Shalev-Shwartz, S., Shammah, S., Shashua, A.: On a formal model of safe and scalable self-driving cars. arXiv preprint:1708.06374 (2017)
32. Shoukry, Y., Nuzzo, P., Sangiovanni-Vincentelli, A.L., Seshia, S.A., Pappas, G.J., Tabuada, P.: SMC: satisfiability modulo convex programming. Proc. of IEEE **106**(9), 1655–1679 (2018)
33. Söntges, S., Althoff, M.: Computing possible driving corridors for automated vehicles. In: Proc. of the IEEE Intell. Veh. Symp., pp. 160–166 (2017)
34. Söntges, S., Althoff, M.: Computing the drivable area of autonomous road vehicles in dynamic road scenes. IEEE Trans. Intell. Transp. Syst. **19**(6), 1855–1866 (2018)
35. Vasirani, M., Ossowski, S.: A market-inspired approach for intersection management in urban road traffic networks. J. Artif. Intell. Res. **43**, 621–659 (2012)
36. Würsching, G., Althoff, M.: Sampling-based optimal trajectory generation for autonomous vehicles using reachable sets. In: Proc. of the IEEE Int. Conf. Intell. Transp. Syst., pp. 828–835 (2021)
37. Zhou, Y., Maity, D., Baras, J.S.: Timed automata approach for motion planning using metric interval temporal logic. In: Proc. of the Eur. Control Conf., pp. 690–695 (2016)
38. Zhu, F., Ukkusuri, S.V.: A linear programming formulation for autonomous intersection control within a dynamic traffic assignment and connected vehicle environment. Transp. Res. Part C: Emerg. Technol. **55**, 363–378 (2015)

A.4 Provably-Safe Cooperative Driving via Invariably Safe Sets [4]

Summary Automated vehicles are expected to offer enhanced safety compared to human drivers, especially in safe-critical situations. This work addresses the problem of provably-safe motion planning for a group of cooperative vehicles operating in mixed traffic scenarios where human and automated vehicles share the road. We propose a novel motion planning approach that ensures the safety of multiple vehicles for an infinite time horizon even under critical situations.

Our approach is founded on the reachability analysis of automated vehicles and the concept of so-called *invariably safe sets* (ISSs): An ISS of a vehicle represents a set of states of the vehicle that allows it to remain safe for an infinite time horizon, regardless of the future behaviors of other traffic participants captured within a set-based prediction. We compute the so-called *safe maneuver corridors* of a vehicle from its ISSs, each representing a space-time envelope corresponding to either a safe braking or a safe evasive maneuver. Through negotiating the safe maneuver corridors of the cooperating vehicles, potential conflicts between their ISSs are efficiently and effectively resolved. Consequently, each vehicle receives its negotiated ISS, which serves as its target set for motion planning. The proposed method (1) can be integrated with arbitrary motion planners that accept position and velocity constraints for the generation of low-level trajectories, (2) ensures that all trajectories planned within the ISSs of the vehicles are safe for an infinite time horizon, and (3) is computationally efficient and real-time capable.

The benefits and applicability of the approach are demonstrated with a highway scenario, a roundabout scenario, and a safety-critical scenario, all from the CommonRoad benchmark suite. We show that our approach ensures the overall safety of the cooperating vehicles while not overly restricting their set of possible trajectories.

Contributions of E. I. L. E. I. L. developed the idea of the research (together with C. P. and M. A.); E. I. L. designed, conducted, and evaluated the experiments (together with C. P.); E. I. L. wrote the article (together with C. P. and M. A.).

Conference article The accepted version of the article is reprinted. The final version of record is available at `https://doi.org/10.1109/IV47402.2020.9304581`.

Provably-safe Cooperative Driving via Invariably Safe Sets

Edmond Irani Liu, Christian Pek, and Matthias Althoff

***Abstract*— We address the problem of provably-safe cooperative driving for a group of vehicles that operate in mixed traffic scenarios, where both autonomous and human-driven vehicles are present. Our method is based on *Invariably Safe Sets* (ISSs), which are sets of states that let each of the cooperative vehicles remain safe for an infinite time horizon. The potential conflicts between the ISSs of a group of cooperative vehicles are resolved by examining and negotiating their *Safe Maneuver Corridors*. As a result, each vehicle obtains its negotiated ISS, which is used as target sets for motion planning. We demonstrate the applicability and benefits of our method on various traffic scenarios from the CommonRoad benchmark suite.**

I. Introduction

Cooperative motion planning of autonomous vehicles offers a great potential to maximize road safety and passenger comfort. As cooperative planners generate trajectories for multiple vehicles, they can more easily find feasible and better solutions for the group, while considering the individual goals of each vehicle. Cooperative planning is particularly challenging in mixed traffic since all motions need to be collision-free with respect to the unknown future motion of other non-cooperative traffic participants.

Invariably Safe Sets (ISSs) [1] are sets of states which ensure that autonomous vehicles can remain safe for an infinite time horizon. If the ISS of a vehicle is used as its target set for motion planning, one can always find motions that do not cause collisions for an infinite time horizon. The application of ISSs to cooperative planning is an auspicious method to provide strict safety guarantees; however, there is no existing method for negotiating ISSs between cooperative vehicles.

A. Related Work

We review existing work on cooperative motion planning and safety verification techniques.

a) Cooperative planning: Cooperative planning methods that are most relevant to our work fall into the reservation-based category [2]. Reservation-based methods ensure that each cooperative vehicle reserves free-space in the environment to plan trajectories. For instance, an intersection manager is presented in [3], [4], which represents lanes as a set of disjunct tiles. These tiles are allocated to vehicles by a first-come-first-serve protocol, and no tile is allocated to more than one vehicle at a time. The reservation-based method was extended in [5] for better performance in mixed traffic. Since first-come-first-serve allocation protocols may be inefficient in traffic scenarios with multiple vehicles, the allocation process has also been carried out using auctions [6]–[9], in which vehicles bid for certain tiles. This bidding process lets one to distribute tiles for maximized usage of the road. However, modeling the environment through tiles often emanates to a large overhead, since usually, only a small percentage of the tiles raise conflicts. Some methods identify possible conflicting points or regions, and allocate these to vehicles if any conflict is detected [10]–[13].

Instead of tiles, some works allocate moving space-time corridors with a predefined behavior, e.g., lane following or lane changing; vehicles that receive these corridors must adopt their behaviors accordingly [14], [15]. As a result, vehicles plan different maneuvers depending on possible future conflicts. In [16], [17], the authors devised an efficient and explicit space-time reservation protocol for cooperative maneuver planning, with which the vehicles broadcast requests on demand.

Despite the mentioned promising results, they cannot ensure safety in arbitrary traffic situations. Yet, this requirement is vital to enable cooperative driving with a high degree of autonomy.

b) Formal verification: Logical reasoning is a means to verify whether a given trajectory fulfills certain safety properties, which are often specified using higher-order logic. For instance, the safety of lane change maneuvers of autonomous vehicles is verified in [18], [19]. Moreover, safely following a preceding vehicle is verified in [20]. Although logical reasoning methods can ensure properties such as safety, the specified logical formulas are often complicated and user-defined. Also, they usually have to be adjusted as new traffic situations arise.

Reachability analysis [21] can be used to verify the safety of trajectories in arbitrary traffic situations. A reachable set is a set of states that a traffic participant can reach over time by starting from an initial set of states and considering all possible control inputs. Planned trajectories of the autonomous vehicle are safe if they do not intersect with predicted unsafe states [22]–[24]. Even though the reachability analysis lets one to consider the set of all possible trajectories of other traffic participants [25] for collision detection, no alternative safe trajectory is returned if a trajectory under examination is regarded as unsafe.

Sets of safe states can be used to ensure that trajectories planned within them remain safe beyond the planning horizon. For instance, trajectories are not allowed to contain inevitable collision states [26], [27], which are states in which the vehicles will eventually collide. In contrast to inevitable collision states, a collision-free trajectory exists

All authors are with the Department of Informatics, Technical University of Munich, 85748 Garching, Germany.
`edmond.irani@tum.de, christian.pek@tum.de, althoff@tum.de`

for states in control invariant sets [28], [29].

B. Contributions

In this work, we propose the first cooperative planning method for autonomous vehicles that incorporates ISSs. In particular, our method

1) provides a mechanism to distribute ISSs to a group of cooperative vehicles;
2) can integrate arbitrary planning algorithms to generate individual trajectories;
3) ensures that all trajectories planned within the ISSs are safe for an infinite time horizon; and
4) is computationally efficient and works in mixed traffic situations.

The rest of this work is organized as follows: In Sec. II, we introduce the necessary preliminaries. Subsequently, we present our solution for cooperative motion planning of autonomous vehicles using ISSs in Sec. III. Following that, we demonstrate the benefits of our method in Sec. IV with exemplary scenarios from the CommonRoad[1] benchmark suite [30]. Finally, we draw conclusions in Sec. V.

II. Preliminaries

In this section, we introduce our system, reachable sets, and invariably safe sets.

A. System Description

We represent the set of considered road lanes as $\mathcal{L} := \{L_1, L_2, \dots, L_M\}$, where each lane L_m possesses its curvilinear coordinate system that takes the centerline as the reference path. Note that we perform all computations in this work in these coordinate systems. The adoption of such a coordinate system expedites the formulation of maneuvers from the ego view of the vehicles, e.g., lane-following, stopping at an intersection, and preventing driving backward. Let V_n be the n-th vehicle in a cooperative group, $\mathcal{V} := \{V_1, V_2, \dots, V_N\}$, where $n \in \mathcal{N} := \{1, 2, \dots, N\}$. We use $\square_n$ to denote a variable of V_n, and $\square_m$ a computation performed in L_m. Also, we introduce the notations $\underline{\square}$ and $\overline{\square}$ to specify the minimal and the maximal possible value of a variable. Furthermore, the notation $[\square_n]_1^N = [\square_1, \dots, \square_N]$ compactly represents a list of variables $\square_n$ of vehicles in $\mathcal{V}$. The system dynamics of V_n is

$$\dot{x}_n(t) = f_n(x_n(t), u_n(t)), \tag{1}$$

where $x_n(t) \in \mathcal{X}_n$ is a state in configuration space, $u_n(t) \in \mathcal{U}_n$ is an input in input space, and t is the time. Given an initial state $x_n(t_0)$ at the initial time t_0, and an input trajectory $u_n([t_0, t])$, the solution to (1) at time t is denoted as $\chi_n(t; x_n(t_0), u_n([t_0, t]))$. The state of vehicle V_n, in lane L_m, is modeled as $x_n(t) = (s_n, d_n, v_n) \in \mathbb{R}^3$, where s_n, d_n, and v_n are the longitudinal position, lateral position, and the velocity of V_n, respectively. Finally, the occupancies of a set of static and dynamic obstacles $\mathcal{E}(t)$ at time t are represented by $\mathcal{O}_{\mathcal{E}}(t) \subset \mathbb{R}^2$.

[1]https://commonroad.in.tum.de/

B. Reachable Sets

Definition 1 (Reachable Set [31]): *The reachable set $\mathcal{R}$ of a system is defined as the set of all states that can be reached at a given time t, starting from an initial set of states $\mathcal{X}_n^0 \subseteq \mathcal{X}_n$.*

During the propagation of the reachable set of vehicle V_n, we remove the set of states whose occupancy is overlapping with $\mathcal{O}_{\mathcal{E}}(t)$. Let $\mathcal{X}_n^{\mathtt{CF}}(t) = \mathcal{X}_n \setminus \mathcal{O}_{\mathcal{E}}(t)$ be the maximal set of states which are collision-free at time t, the collision-free reachable set at time t starting from $\mathcal{X}_n^0$ is

$$\begin{aligned}\mathcal{R}_n(\mathcal{X}_n^0, t) := \Big\{ \chi_n(t; x_n(t_0), u_n([t_0, t])) \Big| x_n(t_0) \in \mathcal{X}_n^0,\\ \forall \tau \in [t_0, t] : u_n(\tau) \in \mathcal{U}_n,\\ \chi_n(\tau; x_n(t_0), u_n([t_0, \tau])) \in \mathcal{X}_n^{\mathtt{CF}}(\tau) \Big\}.\end{aligned}$$

Definition 2 (Drivable Area): *Given that the operator $\mathrm{proj}(\cdot)$ returns the positions (s, d) of a given state $x_n(t)$, the drivable area $\mathcal{D}_n(t)$ of vehicle V_n at time t is defined as the projection of $\mathcal{R}_n(\mathcal{X}_n^0, t)$, thus:*

$$\begin{aligned}\mathcal{D}_n(t) :=& \,\mathrm{proj}(\mathcal{R}_n(\mathcal{X}_n^0, t))\\ =& \Big\{ \mathrm{proj}(x_n(t)) \Big| x_n(t) \in \mathcal{R}_n(\mathcal{X}_n^0, t) \Big\}.\end{aligned}$$

The occupancies $\mathcal{O}_E(t)$ of obstacles $E \in \mathcal{E}(t)$ required for the computation of reachable sets and ISSs are predicted with the help of SPOT [25], which is a set-based occupancy prediction tool that captures all possible future movements of other traffic participants.

C. Invariably Safe Sets

Just being in the collision-free states $\mathcal{X}_n^{\mathtt{CF}}(t)$ does not ensure the safety of the vehicle. Consider the case that a vehicle is driving with a high velocity towards a static obstacle in a near distance. Although it is collision-free at the current time t, it may eventually collide with the obstacle at time $t + t_\epsilon$, $t_\epsilon > 0$. To formally ensure its safety, we, therefore, require the vehicle to be in an invariably safe state [1]. An invariably safe state is defined recursively: a state is deemed invariably safe if a collision-free trajectory ending at another invariably safe state exists.

Definition 3 (Invariably Safe Sets): *The invariably safe set $\mathcal{S}_n(t)$ of V_n at time t is a set of invariably safe states that allows V_n to be safe for an infinite time horizon, and is defined as:*

$$\begin{aligned}\mathcal{S}_n(t) := \Big\{ x_n(t) \in \mathcal{X}_n^{\mathtt{CF}}(t) \Big| \forall \tau > t, \exists u_n([t, \tau]) :\\ \chi_n(\tau; x_n(t), u_n([t, \tau])) \in \mathcal{X}_n^{\mathtt{CF}}(\tau) \Big\}.\end{aligned}$$

In most cases, determining the maximal ISS of V_n is a computationally intensive task. Nonetheless, its tight under-approximation can be efficiently derived using safe braking distances [32] and safe evasive distances [33]. These distances ensure the safety of the ego vehicle even if the preceding obstacles and the obstacles in the adjacent lanes brake or accelerate with their maximal capability.

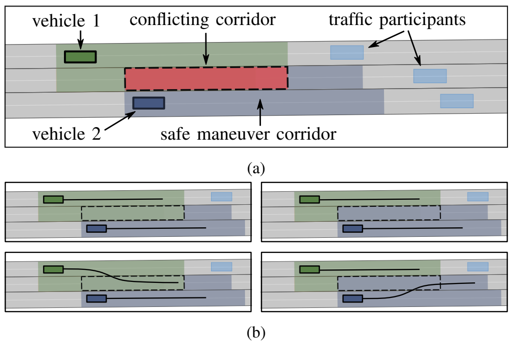


Fig. 1: Overview of our method. (a) Safe maneuver corridors are derived based on individual ISS, and the conflicts between them are examined. (b) Possible allocation plans to resolve the conflict, and exemplary trajectories of vehicles from the set of their permissible trajectories.

Proposition 1 (Under-Approximation of $\mathcal{S}_n(t)$): The union of the set $\mathcal{S}^{\mathtt{b}}_n(t)$ of states respecting safe braking distances [32] and the set $\mathcal{S}^{\mathtt{e}}_n(t)$ of states respecting safe evasive distances [33] at time t is a tight under-approximation of $\mathcal{S}_n(t)$, i.e., $\mathcal{S}^{\mathtt{b}}_n(t) \cup \mathcal{S}^{\mathtt{e}}_n(t) \subset \mathcal{S}_n(t)$. □

Proof: see [1, Sec. III-C]. ■

III. Methodology

We strive to allocate ISSs to a group of cooperative vehicles when conflicts between their individual ISS are detected. It is ensured that safe motions exist when a vehicle is within its ISS [1]. Specifically, the ISSs can be used as target sets for motion planning in two different ways:

1) Solely used in the last time step of the planning cycle to lift safety of the group to an infinite time horizon;
2) Used in every time step to guide motion planners into states for which a vehicle can continue safe motions.

Fig. 1 shows an overview of our method, which can be divided into three major steps:

1) Computation of ISSs for all cooperative vehicles in a group: the computation of the ISS of a vehicle is in turn based on its reachable set and the set-based occupancy prediction of non-cooperative traffic participants [25].
2) Derivation of safe maneuver corridors: to detect the potential conflicts between the ISSs of the vehicles, we derive their safe braking corridors and safe evasive maneuver corridors (hereafter collectively abbreviated as *corridors*) based on their individual ISS. Here, conflicting corridors indicate conflicts of ISSs.
3) Negotiation of conflicting corridors: a negotiation mechanism is established to resolve the detected conflicts of corridors, such that in the end, every vehicle receives a negotiated ISS.

We demand that for each replanning cycle with a replanning duration of t_r, all computations are carried out up to a future time horizon t_h, where $t_r < t_h$. While reaching t_r, the environmental information is updated, and a new round of negotiation within the group is carried out. Violation of such time constraints does not result in unsafe behavior of the vehicles, as their individual fail-safe trajectories [34] are also computed. We perform all computations at discrete time steps $k \geq 0$, which correspond to points in time $t_k = k\Delta t$, where $\Delta t \geq 0$ is a predefined time duration. Subsequently, we assume that all computations are performed for the last time step k of a planning cycle. Alg. 1 describes the procedure of the generation of negotiated ISSs for vehicles. We give a detailed explanation of the algorithm below.

Algorithm 1 Generation of Negotiated ISSs

```
Inputs: $[\mathcal{R}_n(k-1)]_1^N$, $\mathcal{E}_n(k-1)$, $\mathcal{L}$
Output: $[\mathcal{S}^{\mathtt{N}}_n]_1^N$
1: $\mathcal{O}_{\mathcal{E}} \leftarrow \textsc{OccupancyPrediction}(\mathcal{E}_n(k-1))$
2: $[\mathcal{R}_n]_1^N \leftarrow \textsc{ReachableSet}([\mathcal{R}_n(k-1)]_1^N, \mathcal{O}_{\mathcal{E}})$
3: $[\mathcal{S}_n]_1^N \leftarrow \textsc{ISS}([\mathcal{R}_n]_1^N, \mathcal{L}, \mathcal{O}_{\mathcal{E}})$
4: $[\mathcal{C}^{\mathtt{U}}_n]_1^N \leftarrow \textsc{SafeManeuverCorridor}([\mathcal{S}_n]_1^N)$
5: $\mathcal{C}^{\mathtt{C}} \leftarrow \textsc{ConflictingCorridor}([\mathcal{C}^{\mathtt{U}}_n]_1^N)$
6: $P^* \leftarrow \textsc{OptimalAllocation}([\mathcal{S}_n]_1^N, \mathcal{C}^{\mathtt{C}})$
7: $[\mathcal{S}^{\mathtt{N}}_n]_1^N \leftarrow \textsc{NegotiatedISS}([\mathcal{S}_n]_1^N, P^*)$
8: return $[\mathcal{S}^{\mathtt{N}}_n]_1^N$
```

Algorithm 2 Computation of ISS

```
1: function ISS($[\mathcal{R}_n]_1^N$, $\mathcal{L}$, $\mathcal{O}_{\mathcal{E}}$)
2:     for $n \leftarrow 1$ to $N$ do
3:         $\mathcal{D}_n \leftarrow \textsc{Projection}(\mathcal{R}_n)$
4:         $\mathcal{L}^{\mathtt{R}}_n \leftarrow \textsc{ReachableLanes}(\mathcal{D}_n, \mathcal{L})$
5:         $\mathcal{S}^{\mathtt{b}}_n \leftarrow \textsc{SafeBrakingSet}(\mathcal{D}_n, \mathcal{L}^{\mathtt{R}}_n, \mathcal{O}_{\mathcal{E}})$
6:         $\mathcal{S}^{\mathtt{e}}_n \leftarrow \textsc{SafeEvasiveSet}(\mathcal{D}_n, \mathcal{L}^{\mathtt{R}}_n, \mathcal{O}_{\mathcal{E}})$
7:         $\mathcal{S}_n \leftarrow \mathcal{S}^{\mathtt{b}}_n \cup \mathcal{S}^{\mathtt{e}}_n$
8:     end for
9:     return $[\mathcal{S}_n]_1^N$
10: end function
```

A. Computation of Reachable Sets (Alg. 1, line 2)

The dynamics of vehicle V_n are modeled as two double integrators with limited velocities and accelerations in the s_n and d_n directions:

$$\ddot{s}_n(t) = a_{n,s}(t), \quad \ddot{d}_n(t) = a_{n,d}(t), \tag{2a}$$

$$\underline{v}_{n,s} \leq v_{n,s}(t) \leq \overline{v}_{n,s}, \; \underline{v}_{n,d} \leq v_{n,d}(t) \leq \overline{v}_{n,d}, \tag{2b}$$

$$|a_{n,s}(t)| \leq \overline{a}_{n,s}, \quad |a_{n,d}(t)| \leq \overline{a}_{n,d}. \tag{2c}$$

Since model (2) captures the real vehicle behavior, we can use this simple model to efficiently prove safety. The reachable set of V_n, denoted by $\mathcal{R}_n$, is computed as in [31].

B. Computation of ISSs (Alg. 1, line 3)

The computation of individual ISSs of cooperative vehicles are based on their own reachable sets, which is explained as follows (see Alg. 2).

1) Projection: The drivable area of vehicle V_n is $\mathcal{D}_n := \mathrm{proj}(\mathcal{R}_n)$ (cf. Def. 2).

2) Reachable Lanes: $\mathcal{D}_n$ only overlaps with a subset of $\mathcal{L}$, which we refer to as the reachable lanes of V_n, given by:

$$\mathcal{L}^{\mathtt{R}}_n := \{L \in \mathcal{L} | \mathcal{D}_n \cap L \neq \emptyset\}.$$

3) ISSs: The ISS $\mathcal{S}_n$ is computed for every vehicle V_n considering $\mathcal{L}^{\mathsf{R}}_n$. Each of the lanes $L^{\mathsf{R}}_{n,m} \in \mathcal{L}^{\mathsf{R}}_n$ can be split into smaller segments $\mathcal{G}^{i,j}_{n,m}$ which are delimited by different pairs of consecutive obstacles $E_i, E_j \in \mathcal{E}$ located in the lane. We shrink these segments in the driving direction by the half-length of V_n, denoted by $l_n/2$, to accommodate for the occupancy of V_n (see Fig. 2). For simplicity, we omit the superscripts i and j in the following computations, and assume that the occupancies of the obstacles are all within $L^{\mathsf{R}}_{n,m}$. We obtain the drivable area $\mathcal{D}_{n,m}$ that overlaps with the segment $\mathcal{G}_{n,m}$, and its minimal and maximal values along the longitudinal and lateral directions as shown in Fig. 2:

$$
\begin{aligned}
\mathcal{D}_{n,m} &:= \mathcal{D}_n \cap \mathcal{G}_{n,m},\\
[\underline{s}_{n,m}, \overline{s}_{n,m}] &= \mathrm{lon}(\mathcal{D}_{n,m}),\\
[\underline{d}_{n,m}, \overline{d}_{n,m}] &= \mathrm{lat}(\mathcal{D}_{n,m}).
\end{aligned}
$$

where the operators $\mathrm{lon}(\cdot)$ and $\mathrm{lat}(\cdot)$ return the longitudinal and lateral positions $[\underline{s}, \overline{s}]$ and $[\underline{d}, \overline{d}]$ of the given element, respectively.

In the following, we use $q \in \{\mathsf{b}, \mathsf{e}\}$ as the superscript for the sets related to braking and evasive maneuvers. Let $\Delta^{\mathsf{b}}_s(v, E_j)$ be the safe braking distance [32], and $\Delta^{\mathsf{e}}_s(v, d, E_j)$ the safe evasive distance [33], both for a preceding obstacle E_j. We also require the safe braking and evasive distances for a following obstacle E_i, which we denote by $\Delta^{\mathsf{b}}_s(v, E_i)$ and $\Delta^{\mathsf{e}}_s(v, d, E_i)$, respectively. The sets $\mathcal{S}^{\mathsf{b}}_{n,m}$ and $\mathcal{S}^{\mathsf{e}}_{n,m}$ (recall Prop. 1) of vehicle V_n in segment $\mathcal{G}_{n,m}$ are obtained as presented in [1, Alg. 1]:

$$
\begin{aligned}
\mathcal{S}^{\mathsf{b}}_{n,m} := \Big\{ (s,d,v) \Big| \, & s \in [\underline{s}_{n,m}, \overline{s}_{n,m}], \\
& d \in [\underline{d}_{n,m}, \overline{d}_{n,m}], v \in [\underline{v}_{n,s}, \overline{v}_{n,s}], \\
& \forall (s_i, d_i) \in \mathcal{O}_{E_i} : s \geq s_i + \Delta^{\mathsf{b}}_s(v, E_i), \\
& \forall (s_j, d_j) \in \mathcal{O}_{E_j} : s \leq s_j - \Delta^{\mathsf{b}}_s(v, E_j) \Big\},
\end{aligned} \tag{3}
$$

$$
\begin{aligned}
\mathcal{S}^{\mathsf{e}}_{n,m} := \Big\{ (s,d,v) \Big| \, & s \in [\underline{s}_{n,m}, \overline{s}_{n,m}], \\
& d \in [\underline{d}_{n,m}, \overline{d}_{n,m}], v \in [\underline{v}_{n,s}, \overline{v}_{n,s}], \\
& \forall (s_i, d_i) \in \mathcal{O}_{E_i} : s \geq s_i + \Delta^{\mathsf{e}}_s(v, d, E_i), \\
& \forall (s_j, d_j) \in \mathcal{O}_{E_j} : s \leq s_j - \Delta^{\mathsf{e}}_s(v, d, E_j), \\
& \forall \tau \in [0, \Delta^{\mathsf{e}}_t], \exists d' : (s + v\tau, d', v) \in \mathcal{S}^{\mathsf{b}}_{n,m'}(\lceil \tfrac{\tau}{\Delta t} \rceil) \Big\},
\end{aligned} \tag{4}
$$

where m' denotes the index of an adjacent lane of $L^{\mathsf{R}}_{n,m}$ into which the vehicle evades while performing the evasive maneuver, and Δ^{e}_t the evasive time required to fully enter the adjacent lane $L_{m'}$. The velocity constraints $[\underline{v}_{n,s}, \overline{v}_{n,s}]$ are extracted from the reachable set $\mathcal{R}_n$. Refer to [1, Alg. 1] for the derivation of these sets. The ISS of V_n on lanes $\mathcal{L}^{\mathsf{R}}_n$ is defined as:

$$
\mathcal{S}_n := \bigcup_{m,q} \mathcal{S}^q_{n,m}. \tag{5}
$$

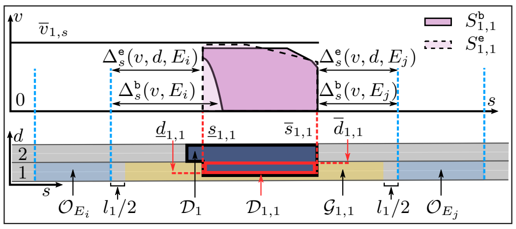


Fig. 2: Relation of various components for ISS computation. The result is shown for a vehicle with $n = 1$, in lane $m = 1$. Lane IDs are labeled at the beginning of the lanes.

C. *Computation of Safe Maneuver Corridors (Alg. 1, line 4)*

We incorporate the set-based occupancy prediction with respect to non-cooperative traffic participants. The potential future collisions between the cooperative vehicles can be detected by examining the future space and time required by the cooperative vehicles to perform safe braking or safe evasive maneuvers, starting from their current ISSs, which we refer to as their *Safe Maneuver Corridors*.

Definition 4 (Safe Maneuver Corridors): *The safe maneuver corridors of vehicle V_n are segments of lanes that V_n requires to perform either a safe braking or a safe evasive maneuver, starting from a state in its ISS.*

A corridor derived from an invariably safe state $(s, d, v) \in \mathcal{S}^q_{n,m}$ in lane L_m is represented by a three-element tuple $(\underline{s}, \overline{s}, k_h)$, which indicates the required position interval $[\underline{s}, \overline{s}]$ in a lane and the time horizon k_h.

For the braking maneuvers, $\Delta^{\mathsf{b}}_t(v)$ and $\Delta^{\mathsf{b}}_s(v)$ are the time required for the vehicle to fully brake, and the distance traveled while braking, respectively; for the evasive maneuvers, we analogously use $\Delta^{\mathsf{e}}_t(d)$ and $\Delta^{\mathsf{e}}_s(v)$. These are given by:

$$
\Delta^{\mathsf{b}}_t(v) = \delta^{\mathsf{b}} + \tfrac{v}{\overline{a}_{n,s}}, \quad \Delta^{\mathsf{b}}_s(v) = \delta^{\mathsf{b}} v + \tfrac{v^2}{2\overline{a}_{n,s}}, \tag{6a}
$$

$$
\Delta^{\mathsf{e}}_t(d) = \delta^{\mathsf{e}} + \sqrt{\tfrac{2 d^{\mathsf{e}}_{m'}(d)}{\overline{a}_{n,d}}}, \quad \Delta^{\mathsf{e}}_s(v) = \Delta^{\mathsf{e}}_t v, \tag{6b}
$$

where δ^{b} and δ^{e} denote the reaction times for braking and evading, respectively, and $d^{\mathsf{e}}_{m'}(d)$ the distance to fully enter an adjacent lane $L_{m'}$ of the lane L_m. For brevity, we omit the road curvature; however, one can easily integrate it as in [1, Alg. 1]. We denote the braking and the evasive corridors of V_n for its ISS $\mathcal{S}^q_{n,m}$ in lane L_m by $\mathcal{C}^{\mathsf{b}}_{n,m}$ and $\mathcal{C}^{\mathsf{e}}_{n,m}$, respectively, given by:

$$
\begin{aligned}
\mathcal{C}^{\mathsf{b}}_{n,m} := \Big\{ & (\underline{s}, \overline{s}, k_h) \Big| (s,d,v) \in \mathcal{S}^{\mathsf{b}}_{n,m}, \\
& \underline{s} = s - l_n/2, \overline{s} = s + \Delta^{\mathsf{b}}_s(v) + l_n/2, k_h = \Big\lceil \tfrac{\Delta^{\mathsf{b}}_t(v)}{\Delta t} \Big\rceil \Big\},
\end{aligned} \tag{7}
$$

$$
\begin{aligned}
\mathcal{C}^{\mathsf{e}}_{n,m} := \Big\{ & (\underline{s}, \overline{s}, k_h) \Big| (s,d,v) \in \mathcal{S}^{\mathsf{e}}_{n,m} \cup \mathcal{S}^{\mathsf{e}}_{n,m'}, \\
& \underline{s} = s - l_n/2, \overline{s} = s + \Delta^{\mathsf{e}}_s(v) + l_n/2, k_h = \Big\lceil \tfrac{\Delta^{\mathsf{e}}_t(d)}{\Delta t} \Big\rceil \Big\}.
\end{aligned} \tag{8}
$$

To simplify the identification of the conflicting corridors in the next step, we formulate tuples $\mathcal{C}^{q,\mathsf{B}}_{n,m}$, which has the extremums over each component of $(\underline{s}, \overline{s}, k_h)$ in $\mathcal{C}^q_{n,m}$ as its

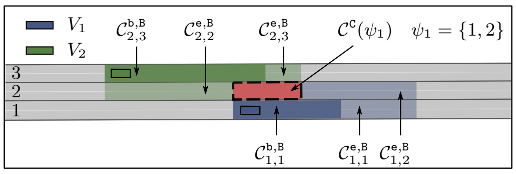

V_1
V_2
$\mathcal{C}^{\mathtt{b},\mathtt{B}}_{2,3}$
$\mathcal{C}^{\mathtt{e},\mathtt{B}}_{2,2}$
$\mathcal{C}^{\mathtt{e},\mathtt{B}}_{2,3}$
$\mathcal{C}^{\mathtt{C}}(\psi_1)$
$\psi_1 = \{1, 2\}$
3
2
1
$\mathcal{C}^{\mathtt{b},\mathtt{B}}_{1,1}$
$\mathcal{C}^{\mathtt{e},\mathtt{B}}_{1,1}$
$\mathcal{C}^{\mathtt{e},\mathtt{B}}_{1,2}$

Fig. 3: Illustration of the (conflicting) safe maneuver corridors. The two vehicles have non-conflicting ISSs in lanes 1 and 3, respectively; however, their corridors intersect in lane 2.

elements (see Fig. 3):

$$\begin{aligned}\mathcal{C}^{q,\mathtt{B}}_{n,m} :=&(\underline{s}', \overline{s}', k_h'), && (9)\\ \underline{s}' =& \min\{\underline{s}|(\underline{s}, \overline{s}, k_h) \in \mathcal{C}^q_{n,m}\},\\ \underline{s}' =& \max\{\overline{s}|(\underline{s}, \overline{s}, k_h) \in \mathcal{C}^q_{n,m}\},\\ k_h' =& \max\{k_h|(\underline{s}, \overline{s}, k_h) \in \mathcal{C}^q_{n,m}\}.\end{aligned}$$

The union of the maneuver corridors of V_n in its reachable lanes $\mathcal{L}^{\mathtt{R}}_n$ over the braking and the evasive maneuvers are:

$$\mathcal{C}^{\mathtt{U}}_n := \bigcup_{m,q}\{\mathcal{C}^{q,\mathtt{B}}_{n,m}\}. \quad (10)$$

D. Identification of Conflicting Corridors (Alg. 1, line 5)

Since conflicting corridors occur between at least two cooperative vehicles, we denote the set of all conflicting subsets of vehicles $\mathcal{V}$ by $\mathcal{I}_{\geq 2}(\mathcal{N})$, which is the power set of $\mathcal{N}$ with a minimum cardinality of two. Each element of $\mathcal{I}_{\geq 2}(\mathcal{N})$ is referred to as a *coalition*, and we denote the i-th coalition in $\mathcal{I}_{\geq 2}(\mathcal{N})$ as ψ_i. The conflicting corridors $\mathcal{C}^{\mathtt{C}}(\psi_i)$ of vehicles within a coalition ψ_i (see Fig. 3), and the tuple of all conflicting corridors $\mathcal{C}^{\mathtt{C}}$ within all coalitions are obtained from

$$\begin{aligned}\mathcal{C}^{\mathtt{C}}(\psi_i) :=& \{(\underline{s}', \overline{s}', k_h')|l, l' \in \psi_i, l \neq l', && (11)\\ &(\underline{s}_1, \overline{s}_1, k_{h,1}) \in \mathcal{C}^{\mathtt{U}}_l, (\underline{s}_2, \overline{s}_2, k_{h,2}) \in \mathcal{C}^{\mathtt{U}}_{l'},\\ &[\underline{s}', \overline{s}'] = [\underline{s}_1, \overline{s}_1] \cap [\underline{s}_2, \overline{s}_2], k_h' = \min(k_{h,1}, k_{h,2})\},\\ \mathcal{C}^{\mathtt{C}} :=& (\mathbf{c}_1, \ldots, \mathbf{c}_j),\ \mathbf{c}_1, \ldots, \mathbf{c}_j \in \mathcal{C}^{\mathtt{C}}(\psi_i), \psi_i \in \mathcal{I}_{\geq 2}(\mathcal{N}). && (12)\end{aligned}$$

E. Negotiation of Conflicting Corridors (Alg. 1, lines 6–7)

The process of finding the optimal allocation plan P^* for the conflicting maneuver corridors is presented in Alg. 3. We explain the main functions as follows:

1) Generation of All Possible Allocation Plans (Alg. 3, line 4): We indicate the j-th element in $\mathcal{C}^{\mathtt{C}}$, which has a conflict within coalition ψ_i, as $\mathbf{c}_j(\psi_i^{(j)})$. A possible allocation of the conflicting corridors is referred to as an allocation plan P, which is a tuple $(P^{(1)}, \ldots, P^{(j)})$, with $P^{(j)}$ representing the index of the winning vehicle of corridor $\mathbf{c}_j(\psi_i^{(j)})$. The set $\mathcal{P}$ of all possible allocation plans P of the conflicting corridors is composed of the Cartesian product of their coalitions $\psi_i^{(j)}$. As an example, for $\mathcal{C}^{\mathtt{C}} = \{\mathbf{c}_1(\psi_1^{(1)}), \mathbf{c}_2(\psi_3^{(2)})\}$, where $\psi_1 = \{1, 2\}$ and $\psi_3 = \{1, 3\}$, $\mathcal{P} = \psi_1 \times \psi_3 = \{(1, 1), (1, 3), (2, 1), (2, 3)\}$ (see Fig. 4).

Algorithm 3 Finding Optimal Allocation Plan P^*

```
1: function OptimalAllocation($[\mathcal{S}_n]_1^N$, $\mathcal{C}^{\mathtt{C}}$)
2:   $P^* \leftarrow \emptyset$,  $J^* \leftarrow \infty$ ▷ initialization of optimal plan and cost
3:   if $\mathcal{C}^{\mathtt{C}} \neq \emptyset$ then
4:     $\mathcal{P} \leftarrow$ AllPossiblePlans($\mathcal{C}^{\mathtt{C}}$)
5:     for $P \in \mathcal{P}$ do
6:       $\mathcal{Z}(P) \leftarrow \emptyset$
7:       for $n \in \mathcal{N}$ do
8:         $\mathcal{S}^{\mathtt{P}}_n(P) \leftarrow$ ISSwithPlan($\mathcal{S}_n$, $P$)
9:         $\mathcal{Z}(P) \leftarrow \mathcal{Z}(P) \bigcup \mathrm{vol}(\mathcal{S}^{\mathtt{P}}_n(P))$
10:      end for
11:      $J(P) \leftarrow$ Cost($\mathcal{Z}(P)$, $[\mathcal{S}^{\mathtt{P}}_n(P)]_1^N$)          ▷ see (15)
12:      if $J(P) < J^*$ then
13:        $P^* \leftarrow P$,  $J^* \leftarrow J(P)$
14:      end if
15:    end for
16:  end if
17:  return $P^*$
18: end function
```

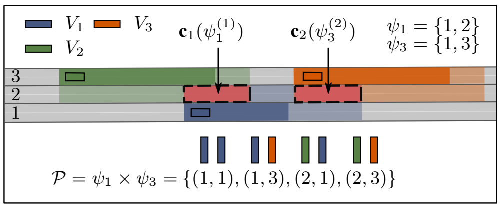

V_1
V_3
V_2
$\mathbf{c}_1(\psi_1^{(1)})$
$\mathbf{c}_2(\psi_3^{(2)})$
$\psi_1 = \{1, 2\}$
$\psi_3 = \{1, 3\}$
3
2
1
$\mathcal{P} = \psi_1 \times \psi_3 = \{(1, 1), (1, 3), (2, 1), (2, 3)\}$

Fig. 4: Illustration of all possible allocation plans.

2) ISSs with respect to plan P (Alg. 3, line 8): We construct a set of corridors that are not allocated to vehicle V_n according to plan P by:

$$\begin{aligned}\mathcal{C}^{\mathtt{L}}_n(P) := \big\{&(\underline{s}, \overline{s}, k_h)|\psi_i^{(j)} \cap n \neq \emptyset, && (13)\\ &n \neq P^{(j)}, (\underline{s}, \overline{s}, k_h) \in \mathcal{C}^{\mathtt{C}}(\psi_i^{(j)})\big\}.\end{aligned}$$

Some of the states in the previously-computed ISS of V_n should be removed as their maneuver corridors intersect with $\mathcal{C}^{\mathtt{L}}_n(P)$. The maximal subset of the ISS $\mathcal{S}^{\mathtt{P}}_n(P) \subseteq \mathcal{S}_n$ that is still safe with respect to plan P is obtained through:

$$\mathrm{cor}(\mathcal{S}^{\mathtt{P}}_n(P)) = \mathrm{cor}(\mathcal{S}_n) \setminus \mathcal{C}^{\mathtt{L}}_n(P),$$

with the operator $\mathrm{cor}(\cdot)$ returning the set of corridors $\mathcal{C}^{\mathtt{U}}_n$ (cf. (7)–(10)) of a given input ISS.

3) Cost Function (Alg. 3, lines 6–14): We aim for a fair allocation of the conflicting corridors. To this end, we compute the variance of the volumes of $\mathcal{S}^{\mathtt{P}}_n(P)$, which are returned by the operator $\mathrm{vol}(\cdot)$, of vehicles $\mathcal{V}$ for each plan. Let $\mathcal{Z}(P) := \{Z_1, Z_2, \ldots, Z_N\}$ be the set of volumes Z_n of $\mathcal{S}^{\mathtt{P}}_n(P)$, the variance of $\mathcal{Z}(P)$, which is returned by $\mathrm{var}(\cdot)$, is computed by:

$$\tilde{Z} = \frac{\sum_1^N Z_n}{N}, \ \mathrm{var}(\mathcal{Z}(P)) := \frac{\sum_1^N (Z_n - \tilde{Z})^2}{N}. \quad (14)$$

In the best case $\mathrm{var}(\mathcal{Z}(P)) = 0$, implying that all of the cooperative vehicles receive equally large ISS.

In addition, we reward allocation plans that promote higher overall traffic efficiency, which is reflected by summing the length of the longest maneuver corridors in the longitudinal direction of all cooperative vehicles. The proposed cost function for a plan P has the form of

$$J(P) = \mathrm{var}(\mathcal{Z}(P)) - w \sum_{1}^{N} \max\big\{\, \mathrm{len}(\mathbf{c}) \mid \mathbf{c} \in \mathrm{cor}(\mathcal{S}_n^{\mathtt{P}}(P)) \big\}, \qquad (15)$$

where w is an user-tunable parameter, and the $\mathrm{len}(\cdot)$ operator returns the length $|\overline{s} - \underline{s}|$ of the given corridor. Alternatively, one can also include other terms into the cost function. The plan with the minimal cost is regarded as the optimal allocation plan P^*, and the ISSs of vehicles with respect to P^* is referred to as their *Negotiated ISSs* $\mathcal{S}_n^{\mathtt{N}} := \mathcal{S}_n^{\mathtt{P}}(P^*)$.

F. Incorporating Negotiated ISSs into Every Time Step

To incorporate the negotiated ISSs into every time step of the current planning cycle (cf. Sec. III), for a vehicle V_n that has a non-empty $\mathcal{C}_n^{\mathtt{L}}(P^*, k)$ at time step k, we stipulate that it respects the negotiation result by not entering the corridors $\mathbf{c} \in \mathcal{C}_n^{\mathtt{L}}(P^*, k)$, up to its persisting time step $k + k_h$. This is achieved by removing the occupancies of $\mathbf{c}$ in the computation of the reachable set of V_n in subsequent time steps (cf. Def. 1). Additionally, we require that V_n respects safe distances with regards to $\mathbf{c}$, with their velocity profiles extracted from the winning vehicles at longitudinal positions $\mathrm{lon}(\mathbf{c})$, to ensure safety within the planning cycle. To this end, we formulate a set of phantom obstacles $\mathcal{E}_n^{\mathtt{F}}(k)$, which has $\mathcal{C}_n^{\mathtt{L}}(P^*, k)$ and the persisting corridors $\mathbf{c}$ from previous time steps as its elements:

$$\mathcal{E}_n^{\mathtt{F}}(k) := \mathcal{C}_n^{\mathtt{L}}(P^*, k) \cup \big\{ \mathbf{c} \in \mathcal{E}_n^{\mathtt{F}}(k-1) \big| k \leq \mathrm{per}(\mathbf{c}) \big\},$$

with the operator $\mathrm{per}(\cdot)$ returning the persisting time step of the given corridor.

G. Computational Complexity

Assuming that the set-based occupancy prediction of obstacles and the reachable sets of cooperative vehicles are readily available (see [25], [31] for their respective complexity analysis), the computational complexity of the proposed method for each time step k can be split into three parts:

1) Computation of ISSs: for a vehicle V_n, the complexity of computing its ISS $\mathcal{S}_n$ is $O(N)$, where N denotes the number of considered obstacles in lanes $\mathcal{L}_n^{\mathtt{R}}$.
2) Derivation of Safe Maneuver Corridors: the complexity of computing the safe maneuver corridors of V_n is again $O(N)$.
3) Negotiation of Conflicting Corridors: for a group of N vehicles that collectively access M reachable lanes, in the worst case, they could have conflicts of corridors on all reachable lanes, thus raising a complexity of $O(N^M)$. Nevertheless, the computational tractability can still be ensured, since

TABLE I: PARAMETERS FOR NUMERICAL EXPERIMENTS.

Parameters		**Scenario Identifiers**		
symbol	**unit**	I	II	III
Δt	s	0.1	0.1	0.1
t_h	s	2.5	2.0	2.0
$\overline{v}_{n,s}$	m/s	30.0	30.0	30.0
$\underline{v}_{n,s}$	m/s	0.0	0.0	0.0
$\overline{v}_{n,d}$	m/s	3.0	3.0	4.0
$\underline{v}_{n,d}$	m/s	-3.0	-3.0	-4.0
$\overline{a}_{n,s}$	m/s^2	8.0	8.0	8.0
$\overline{a}_{n,d}$	m/s^2	3.0	3.0	4.0
$\delta^{\mathtt{b}}$	s	0.3	0.3	0.3
$\delta^{\mathtt{e}}$	s	0.3	0.3	0.3

a) the upper bounds of both N and M can be controlled by forming smaller cooperative groups, e.g. using methods as in [35]; and
b) the proposed method can be applied in *anytime* fashion, i.e., the negotiated ISSs with respect to the best plan evaluated up to the time of request is returned.

Alternatively, one can adopt auction-based methods, as in [6], to obtain a sub-optimal allocation of conflicting corridors with polynomial complexity with regard to M.

IV. Evaluation

We demonstrate the applicability of our method on three distinct scenarios. A list of selected parameters for all the cooperative vehicles can be found in Tab. I. We use an optimization-based motion planner [34] to generate trajectories within $\mathcal{S}_n^{\mathtt{N}}(k)$.

A. Scenario I: Highway

Our first scenario illustrates a highway with two communicating vehicles V_1 and V_2 in the presence of two leading other traffic participants (see Fig. 5). V_1 and V_2 face a conflict of maneuver corridors at the very first time step $k = 1$. Our algorithm allocates the conflicting corridor to V_1. As the reachable sets of V_1 progresses to lane 2 at time step $k = 10$, another conflicting corridor arises, which is then allocated to V_2. The negotiated ISSs demand that V_2 keeps driving straight while allowing V_1 to choose freely from keeping its lane or swerving into the center lane. Fig. 6 visualizes the negotiated ISSs and the states of the planned trajectories; as can be seen, the states are all within the negotiated ISSs, thus safety is ensured. Alternatively, one can only use the negotiated ISSs in the last time step, if the safety within the planning cycle is already ensured by the motion planners.

B. Scenario II: Merging

In our second scenario, V_1 and V_2 approach a merging point at a roundabout. Note that for complex road geometries, the maneuver corridors of vehicles are projected onto the intersecting lanes to compute the conflicts (see Fig. 7). V_1 gets the conflicting corridor allocated at time step $k = 6$, and hence, is entitled to keep driving along its lane in the

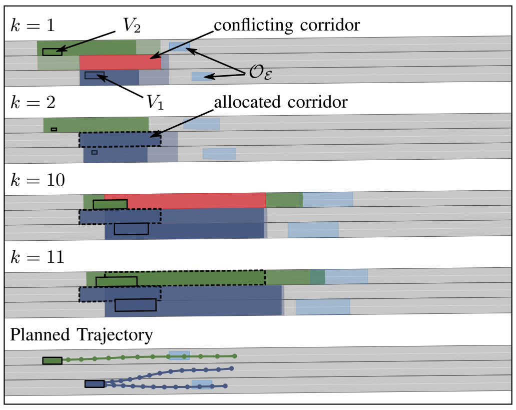
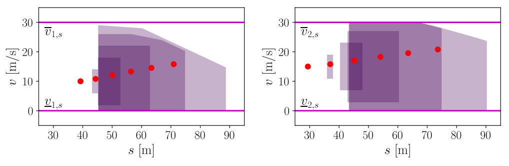


Fig. 5: Scenario I: Highway. The driving direction is from left to right. In the subplots for time step $k = 1$, and the planned trajectory, we show the initial positions of vehicles; in other cases the drivable area. Based on the negotiation result, vehicle V_1 is let to keep or change its lane.



(a) $\mathcal{S}_n^{\mathtt{N}}(k)$ of V_1 on lane 1, keep lane. (b) $\mathcal{S}_n^{\mathtt{N}}(k)$ of V_2 on lane 3, keep lane.

Fig. 6: The negotiated ISSs $\mathcal{S}_n^{\mathtt{N}}(k)$ (with $d \approx 0$ shown) of vehicles for time steps $k = 0$ to 25 with a step size of 5. The red dots indicate the state of the vehicles at each time step.

subsequent time steps. On the other hand, the ISS of V_2 is constrained to remain right in front of the merging point before V_1 passes.

C. Scenario III: Safety-Critical

Our last scenario demonstrates a safety-critical situation (see Fig. 8): a truck driving in the rightmost lane had an accident, causing it to block the two rightmost lanes. V_1 is already too close to the truck to perform a safe braking maneuver. The only way that V_1 can remain safe is to perform an evasive maneuver with the help of vehicle V_2. Even in such a critical situation, our method can determine the correct coordination of V_1 and V_2 to ensure safety. Similar situations can arise due to the presence of unexpected pedestrians, construction sites on roads, etc.

V. Conclusions

In this work, by incorporating ISSs, we propose a novel cooperative planning method that ensures the safety of a group of cooperative vehicles for an infinite time horizon. Our method computes ISS for each cooperative vehicle and derives the negotiated ISSs by resolving conflicts between their safe maneuver corridors. These conflicting corridors are allocated to the vehicles, and they serve as constraints for the propagation of subsequent reachable sets. In various scenarios with mixed-traffic, we demonstrate that our method ensures the overall safety of the cooperative group while not overly restricting the set of possible trajectories for the vehicles.

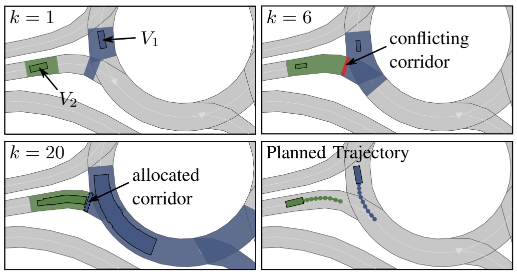


Fig. 7: Scenario II: Merging. The driving direction is indicated by the white arrows on the lanes. In the subplots for time step $k = 1$, and the planned trajectory, we show the initial positions of vehicles; in other cases the drivable area. Based on the negotiation result, vehicle V_1 is prioritized.

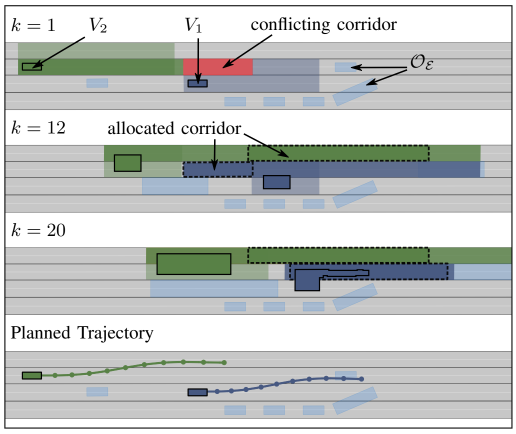


Fig. 8: Scenario III: Safety-Critical. The driving direction is from left to right. In the subplots for time step $k = 1$, and the planned trajectory, we show the initial positions of vehicles; in other cases the drivable area. Based on the negotiation result, both vehicles change lane to ensure the safety.

Acknowledgments

This project was funded by the Deutsche Forschungsgemeinschaft (German Research Foundation) within the Priority Programme SPP 1835 *Cooperative Interacting Automobiles* under grant No. AL 1185/4-2, and within the EU Horizon 2020 Programme *interACT* under grant No. 723395. The authors appreciate the fruitful collaboration with the project partners.

References

[1] C. Pek and M. Althoff, “Efficient computation of invariably safe states for motion planning of self-driving vehicles,” in *Proc. of the IEEE Int. Conf. Intell. Robot. Syst.*, 2018, pp. 3523–3530.

[2] J. Rios-Torres and A. A. Malikopoulos, “A survey on the coordination of connected and automated vehicles at intersections and merging at highway on-ramps,” *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 5, pp. 1066–1077, 2017.

[3] K. Dresner and P. Stone, “Human-usable and emergency vehicle-aware control policies for autonomous intersection management,” in *Workshop Agents Traffic Transp.*, 2006, pp. 17–25.

[4] ——, “Turning the corner: improved intersection control for autonomous vehicles,” in *Proc. of the IEEE Intell. Veh. Symp.*, 2005, pp. 423–428.

[5] G. Sharon and P. Stone, “A protocol for mixed autonomous and human-ooperated vehicles at intersections,” in *Proc. of the Int. Joint Conf. Auton. Agent. Multi-Agent Syst.*, 2017, pp. 151–167.

[6] S. Manzinger and M. Althoff, “Tactical decision making for cooperative vehicles using reachable sets,” in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2018, pp. 444–451.

[7] D. Carlino, S. D. Boyles, and P. Stone, “Auction-based autonomous intersection management,” in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2013, pp. 529–534.

[8] M. Vasirani and S. Ossowski, “A market-inspired approach for intersection management in urban road traffic networks,” *J. Artific. Intell. Res.*, vol. 43, pp. 621–659, 2012.

[9] H. Schepperle and K. Böhm, “Auction-based traffic management: towards effective concurrent utilization of road intersections,” in *Proc. of the IEEE Conf. E-Commerce Technol. and IEEE Conf. Enterprise Comput, E-Commerce, E-Services*, 2008, pp. 105–112.

[10] S. Manzinger and M. Althoff, “Negotiation of drivable areas of cooperative vehicles for conflict resolution,” in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2018, pp. 1–8.

[11] M. W. Levin, H. Fritz, and S. D. Boyles, “On optimizing reservation-based intersection controls,” *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 3, pp. 505–515, 2017.

[12] I. A. Ntousakis, I. K. Nikolos, and M. Papageorgiou, “Optimal vehicle trajectory planning in the context of cooperative merging on highways,” *Transp. Res. Part C Emerg. Technol.*, vol. 71, pp. 464–488, 2016.

[13] F. Zhu and S. V. Ukkusuri, “A linear programming formulation for autonomous intersection control within a dynamic traffic assignment and connected vehicle environment,” *Transp. Res. Part C Emerg. Technol.*, vol. 55, pp. 363–378, 2015.

[14] D. Marinescu, J. Čurn, M. Bouroche, and V. Cahill, “On-ramp traffic merging using cooperative intelligent vehicles: a slot-based approach,” in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2012, pp. 900–906.

[15] D. Marinescu, J. Čurn, M. Slot, M. Bouroche, and V. Cahill, “An active approach to guaranteed arrival times based on traffic shaping,” in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2010, pp. 1711–1717.

[16] M. Nichting, D. Heß, J. Schindler, T. Hesse, and F. Köster, “Explicit negotiation method for cooperative automated vehicles,” in *Proc. of the IEEE Int. Conf. Veh. Electron. Safety*, 2019, pp. 1–7.

[17] D. Heß, R. Lattarulo, J. Pérez, J. Schindler, T. Hesse, and F. Köster, “Fast maneuver planning for cooperative automated vehicles,” in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2018, pp. 1625–1632.

[18] W. Damm, H.-J. Peter, J. Rakow, and B. Westphal, “Can we build it: formal synthesis of control strategies for cooperative driver assistance systems,” *Math. Struct. in Comp. Science*, vol. 23, no. 04, pp. 676–725, 2013.

[19] M. Hilscher, S. Linker, and E.-R. Olderog, “Proving safety of traffic manoeuvres on country roads,” in *Theories of Programming and Formal Methods*, 2013, pp. 196–212.

[20] S. Mitsch, S. M. Loos, and A. Platzer, “Towards formal verification of freeway traffic control,” in *Proc. of the IEEE Conf. Cyber-Physical Syst.*, 2012, pp. 171–180.

[21] I. M. Mitchell, “Comparing forward and backward reachability as tools for safety analysis,” in *Hybrid Systems: Computation and Control*, 2007, pp. 428–443.

[22] M. Althoff and J. M. Dolan, “Online verification of automated road vehicles using reachability analysis,” *IEEE Trans. Robot.*, vol. 30, no. 4, pp. 903–918, 2014.

[23] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, “FaSTrack: a modular framework for fast and guaranteed safe motion planning,” in *Proc. of the IEEE Conf. Decis. Control*, 2017, pp. 1517–1522.

[24] P. Falcone, M. Ali, and J. Sjöberg, “Predictive threat assessment via reachability analysis and set invariance theory,” *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1352–1361, 2011.

[25] M. Koschi and M. Althoff, “SPOT: a tool for set-based prediction of traffic participants,” in *Proc. of the IEEE Intell. Veh. Symp.*, 2017, pp. 1686–1693.

[26] T. Fraichard, “A short paper about motion safety,” in *Proc. of the IEEE Int. Conf. Robot. Autom.*, 2007, pp. 1140–1145.

[27] D. Althoff, J. J. Kuffner, D. Wollherr, and M. Buss, “Safety assessment of robot trajectories for navigation in uncertain and dynamic environments,” *Auton. Robots*, vol. 32, no. 3, pp. 285–302, 2012.

[28] D. Althoff, M. Althoff, and S. Scherer, “Online safety verification of trajectories for unmanned flight with offline computed robust invariant sets,” in *Proc. of the IEEE Int. Conf. Intell. Robot. Syst.*, 2015, pp. 3470–3477.

[29] K. Berntorp, A. Weiss, C. Danielson, and S. Di Cairano, “Automated driving: safe motion planning using positively invariant sets,” in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2017, pp. 1–6.

[30] M. Althoff, M. Koschi, and S. Manzinger, “CommonRoad: composable benchmarks for motion planning on roads,” in *Proc. of the IEEE Intell. Veh. Symp.*, 2017, pp. 719–726.

[31] S. Söntges and M. Althoff, “Computing the drivable area of autonomous road vehicles in dynamic road scenes,” *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 6, pp. 1855–1866, 2018.

[32] A. Rizaldi, F. Immler, and M. Althoff, “A formally verified checker of the safe distance traffic rules for autonomous vehicles,” in *NASA Formal Methods Symposium*, vol. 9690, 2016, pp. 175–190.

[33] C. Pek, P. Zahn, and M. Althoff, “Verifying the safety of lane change maneuvers of self-driving vehicles based on formalized traffic rules,” in *Proc. of the IEEE Intell. Veh. Symp.*, 2017, pp. 1477–1483.

[34] C. Pek and M. Althoff, “Computationally efficient fail-safe trajectory planning for self-driving vehicles using convex optimization,” in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2018, pp. 1447–1454.

[35] C. Frese, J. Beyerer, and P. Zimmer, “Cooperation of cars and formation of cooperative groups,” in *Proc. of the IEEE Intell. Veh. Symp.*, 2007, pp. 227–232.

A.5 CommonRoad-Reach: A Toolbox for Reachability Analysis of Automated Vehicles [5]

Summary Reachability analysis has gained considerable attention and popularity in recent years for automated vehicle applications, including motion planning and safety assurance. While there exist tools for reachability analysis that focus on general-purpose algorithms for formal verification of dynamic systems, a toolbox tailored to automated vehicle-specific applications has yet to be publicly available. This work precisely addresses this gap by providing the open-source CommonRoad-Reach toolbox.

The CommonRoad-Reach toolbox offers several methods for the computation of reachable sets of automated vehicles and the extraction of driving corridors. In comparison to existing toolboxes, CommonRoad-Reach offers the following functionalities and advantages: (1) The integration of two approaches for computing reachable sets, specifically, employing polytopic set propagation and graph-based propagation techniques. (2) The extraction of collision-free driving corridors, which can be utilized as planning constraints for motion planners. (3) Offering Python and C++ implementations of the algorithms, providing both the convenience of prototyping and real-time computation to users. (4) Integration within the CommonRoad benchmark suite, providing a simulation framework, an extensive scenario database, and various tools for motion planning of automated vehicles. We use multiple dynamic traffic scenarios of varied complexity from the CommonRoad benchmark suite to exhibit our toolbox's key features and functionalities. Moreover, we benchmark the performance of our implementations against 100 scenarios.The results indicate that the computation time requires only a fraction of the planning horizon; thus, our toolbox is capable of real-time.

Contributions of E. I. L. E. I. L. developed the idea of the research (together with G. W., M. K., and M. A.); E. I. L. designed, conducted, and evaluated the experiments (together with G. W. and M. K.); E. I. L. wrote the article (together with G. W., M. K., and M. A.).

Conference article The accepted version of the article is reprinted. The final version of record is available at `https://doi.org/10.1109/ITSC55140.2022.9922232`.

Attachment The animation of the evaluations is available at `https://mediatum.ub.tum.de/1662399`.

CommonRoad-Reach: A Toolbox for Reachability Analysis of Automated Vehicles

Edmond Irani Liu*, Gerald Würsching*, Moritz Klischat, and Matthias Althoff

***Abstract*— In recent years, reachability analysis has gained considerable popularity in motion planning and safeguarding of automated vehicles (AVs). While existing tools for reachability analysis mainly focus on general-purpose algorithms for formal verification of dynamical systems, a toolbox tailored to AV-specific applications is not yet available. In this study, we present CommonRoad-Reach, which is a toolbox that integrates different methods for computing reachable sets and extracting driving corridors for AVs in dynamic traffic scenarios. Our toolbox provides a Python interface and an efficient C++ implementation for real-time applications. The toolbox is integrated within the CommonRoad benchmark suite and is available at commonroad.in.tum.de.**

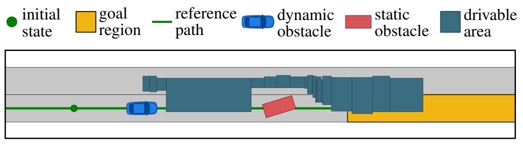


Fig. 1: Exemplary computation result of our toolbox for a simple scenario with a static and dynamic obstacle. Our tool computes the reachable set of the ego vehicle over time considering the initial state and (time-varying) forbidden states. The projection of the reachable set onto the position domain yields a collision-free drivable area shown here for step $k = 28$.

I. Introduction

Compared with human-driven vehicles, highly automated vehicles (AVs) are expected to offer increased road safety and passenger comfort, reduced emissions and travel time. Major challenges, such as strict safety guarantees in critical situations, decision making, and motion planning in small solution space, are yet to be resolved to fully unfold these benefits. Reachability analysis, which determines the set of all reachable states (also referred to as *reachable set*) of a system over time, is a powerful technique to address these challenges. An example of a reachable set is shown in Fig. 1. Despite reachability analysis having been well-researched over the past decades with continuous improvements in scalability and tightness [1], publicly available toolboxes are either not real-time capable for AV-specific applications or do not take into account time-varying forbidden states originating from traffic participants present in the scenario. This study presents a toolbox for computing the reachable sets of AVs for motion planning applications. The toolbox integrates two methods presented in our previous works: polytopic set propagation [2] and graph-based propagation [3].

A. Related Work

1) Reachability analysis for road vehicles: General-purpose approaches for reachability analysis are primarily used for formal verification, i.e., checking whether a system can reach unsafe sets considering system dynamics and constraints, e.g., on input or disturbance bounds. These methods are useful for AVs to verify their motion plans, see, e.g., [4], [5]. However, motion planning or rigorous computation of safety metrics, such as time-to-react [6], requires especially efficient algorithms considering surrounding moving vehicles.

Several intriguing applications of reachability analysis for AVs have recently been proposed in the literature: Reachability analysis can be used to determine the set of states that ultimately result in a collision irrespective of the input of choice [7]. Set-based prediction using reachability analysis has been proposed to capture all possible future behaviors of surrounding traffic participants, which also considers limitations due to the field-of-view and traffic rules [8]–[10]. Online verification techniques that utilize safety layers based on reachability analysis can ensure the safety of motion planners [11]–[15]. A further line of research uses reachable sets to extract possible driving corridors, i.e., spatio-temporal constraints, for motion planning [16], [17]. For this application, set-based techniques are especially well suited in complex scenarios as they do not suffer from discretization effects and are computationally feasible under real-time constraints, in contrast to other methods, such as sampling-based [18] or combinatorial [19] approaches. Driving corridors extracted from reachable sets are integrated with different motion planners in [16], [20]. Furthermore, reachable sets are used to determine specification-compliant planning space [21] and to negotiate conflicting planning space among a group of cooperating vehicles [22].

2) Existing toolboxes for reachability analysis: Recently, several publicly available toolboxes for reachability analysis have been developed. Tools such as Flow* [23] and SpaceEx [24] offer efficient C++ implementations of set representations and reachability algorithms for linear [24] and nonlinear [23] hybrid systems. Although C++-based tools have good performance, their compilation overhead makes them difficult to use for prototyping. Hence, reachability tools written in just-in-time compiled or interpreted languages are desirable; examples include the MATLAB-based tool CORA [25], JuliaReach [26], or the Python-based tool HyLAA [27].

* The first two authors have contributed equally to this work.

All authors are with the Department of Informatics, Technical University of Munich, 85748 Garching, Germany.
{edmond.irani,gerald.wuersching,moritz.klischat,althoff}@tum.de

B. Contributions

Existing toolboxes are either not capable of excluding time-varying forbidden states [23]–[26], [28], or are based on inherently inefficient Hamilton–Jacobi–Bellmann solvers [29]–[32]. Despite the aforementioned applications of reachable sets in recent works, a toolbox implementing those methods is not publicly available. Our open-source toolbox is tailored to AV-specific applications and

- integrates two approaches for computing reachable sets, i.e., using polytopic set propagation and graph-based propagation;
- extracts collision-free driving corridors that can be used as planning constraints for motion planners;
- provides Python and C++ implementations of the algorithms, offering convenient prototyping and real-time computation to the users; and
- is integrated within the CommonRoad[1] benchmark suite [33], which offers a simulation framework, an extensive scenario database, and various tools for motion planning of automated vehicles.

The remainder of this study is structured as follows: Sec. II introduces necessary preliminaries and Sec. III highlights the implementation details of our toolbox. In Sec. IV, we demonstrate the key features of our toolbox in numerical experiments. Lastly, we draw conclusions in Sec. V.

II. Preliminaries

A. System Description

The scenarios in our toolbox are described in the CommonRoad format, which represents the road network using lanelets [34] and models environment elements such as obstacles and traffic signs. We predict the motion of dynamic obstacles using their most-likely trajectories; however, our toolbox is also adaptable to any other prediction method such as set-based prediction [8]. Both the global Cartesian coordinate system and a local curvilinear coordinate system that is aligned with a reference path [34] can be used to compute the reachable set. The possible reference path is the centerline of a lane or a path through the road network leading from the initial state to a specified goal region.

Let us introduce some necessary notations: We denote by $k \in \mathbb{N}_0$ a discrete step corresponding to a continuous time $t_k = k\Delta_t$, with a predefined time increment $\Delta_t \in \mathbb{R}_+$. For a given dynamical system, $\boldsymbol{x}_k \in \mathcal{X}_k \subset \mathbb{R}^4$ represents a state in the state space $\mathcal{X}_k$, and $\boldsymbol{u}_k \in \mathcal{U}_k \subset \mathbb{R}^2$ represents an input in the input space $\mathcal{U}_k$, each at step k. Let $\square$ be a variable, we denote its minimum and maximum values by $\underline{\square}$ and $\overline{\square}$, respectively. Since our toolbox facilitates the computation in Cartesian and curvilinear coordinate systems, we introduce the general subscripts $\zeta \in \{x, s\}$ and $\eta \in \{y, d\}$ to indicate the direction of a variable in the corresponding coordinate frame. Thus, (x, y) denotes the global coordinates in the Cartesian frame, and (s, d) denotes the longitudinal coordinate s and lateral coordinate d in a local curvilinear frame. Computations within the Cartesian coordinate system do not rely on a reference path and handle unstructured scenarios better. In contrast, computations within a curvilinear coordinate system are better suited for structured scenarios with lanes.

We abstract the system dynamics of a hypothetical ego vehicle using a point-mass model. This abstraction ensures that the reachable set of the high-fidelity model is always a subset of that of the simplified model; alternative abstractions can be found in [35], [36]. The state of the system is modeled as $\boldsymbol{x}_k = (p_{\zeta,k}, v_{\zeta,k}, p_{\eta,k}, v_{\eta,k})^\mathsf{T}$, and the system accepts inputs $\boldsymbol{u}_k = (a_{\zeta,k}, a_{\eta,k})^\mathsf{T}$, where p, v, a denote position, velocity, and acceleration, respectively. The discrete-time system dynamics of the ego vehicle is

$$\boldsymbol{x}_{k+1} = \begin{pmatrix} 1 & \Delta_t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta_t \\ 0 & 0 & 0 & 1 \end{pmatrix} \boldsymbol{x_k} + \begin{pmatrix} \frac{1}{2}\Delta_t^2 & 0 \\ \Delta_t & 0 \\ 0 & \frac{1}{2}\Delta_t^2 \\ 0 & \Delta_t \end{pmatrix} \boldsymbol{u}_k. \tag{1}$$

The velocities and accelerations in both directions of the coordinate system are bounded by

$$\underline{v}_\zeta \leq v_{\zeta,k} \leq \overline{v}_\zeta,\ \underline{v}_\eta \leq v_{\eta,k} \leq \overline{v}_\eta, \tag{2a}$$

$$\underline{a}_\zeta \leq a_{\zeta,k} \leq \overline{a}_\zeta,\ \underline{a}_\eta \leq a_{\eta,k} \leq \overline{a}_\eta. \tag{2b}$$

These bounds are chosen to consider the kinematic limitations within the adopted coordinate system, see, e.g., [37]. Note that within the Cartesian coordinate system, we over-approximate Kamm's friction circle with a box; within curvilinear coordinate systems, we assume that the ego vehicle follows the reference path and thus use more conservative parameters for the longitudinal and lateral driving directions.

B. Reachable Set

Let $\boldsymbol{x}_0$ be the initial state and $\boldsymbol{u}_{[0,k]}$ be an input trajectory between steps 0 and k. We use $\chi_k(\boldsymbol{x}_0, \boldsymbol{u}_{[0,k]})$ to represent the solution of (1) at step k. Given the occupancy of the ego vehicle $\mathcal{Q}(\boldsymbol{x}_k) \subset \mathbb{R}^2$ and the set of time-varying occupancies of all obstacles $\mathcal{O}_k \subset \mathbb{R}^2$ at step k, we define the set of forbidden states as $\mathcal{F}_k = \{\boldsymbol{x}_k \in \mathcal{X}_k | \mathcal{Q}(\boldsymbol{x}_k) \cap \mathcal{O}_k \neq \emptyset\}$. In this study, the reachable set of the ego vehicle at step k is defined as the set of states reachable from the initial set of states $\mathcal{X}_0$ while avoiding the set of forbidden states $\mathcal{F}_\tau$ for every step $\tau \in \{0, \ldots, k\}$ [2]:

$$\mathcal{R}_k^*(\mathcal{X}_0) := \Big\{ \chi_k(\boldsymbol{x}_0, \boldsymbol{u}_{[0,k]}) \Big| \exists \boldsymbol{x}_0 \in \mathcal{X}_0, \forall \tau \in \{0, \ldots, k\}, \\ \exists \boldsymbol{u}_\tau \in \mathcal{U}_\tau : \chi_\tau(\boldsymbol{x}_0, \boldsymbol{u}_{[0,\tau]}) \notin \mathcal{F}_\tau \Big\}.$$

Subsequently, we omit $\mathcal{X}_0$ for brevity. Obtaining $\mathcal{R}_k^*$ is in general computationally expensive; therefore, we compute its over-approximation $\mathcal{R}_k$. As a set representation, we choose the union of base sets $\mathcal{R}_k^{(i)}$, $i \in \mathbb{N}$. Each base set $\mathcal{R}_k^{(i)} = \mathcal{P}_{\zeta,k}^{(i)} \times \mathcal{P}_{\eta,k}^{(i)}$ is a Cartesian product of two convex polytopes $\mathcal{P}_{\zeta,k}^{(i)}$ and $\mathcal{P}_{\eta,k}^{(i)}$, which represent the reachable positions and velocities in the (p_ζ, v_ζ) and (p_η, v_η) planes, respectively (Fig. 2a–b). This representation is beneficial compared to other set representations in [1, Tab. 1] since

[1]https://commonroad.in.tum.de/

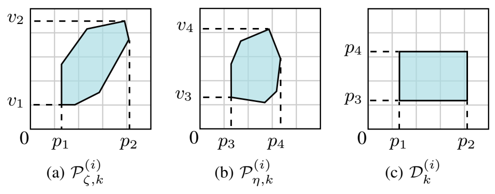

v_2
v_1
0
p_1
p_2
(a) $\mathcal{P}^{(i)}_{\zeta,k}$
v_4
v_3
0
p_3
p_4
(b) $\mathcal{P}^{(i)}_{\eta,k}$
p_4
p_3
0
p_1
p_2
(c) $\mathcal{D}^{(i)}_k$

Fig. 2: Polytopes and drivable area of a base set $\mathcal{R}^{(i)}_k$.

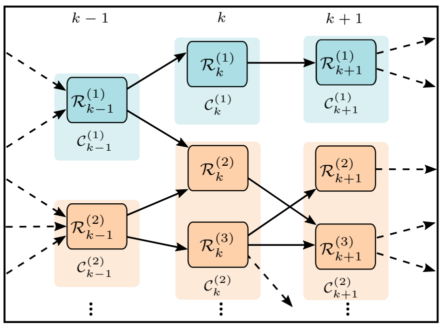

$k-1$
k
$k+1$
$\mathcal{R}^{(1)}_{k-1}$
$\mathcal{C}^{(1)}_{k-1}$
$\mathcal{R}^{(1)}_k$
$\mathcal{C}^{(1)}_k$
$\mathcal{R}^{(1)}_{k+1}$
$\mathcal{C}^{(1)}_{k+1}$
$\mathcal{R}^{(2)}_{k-1}$
$\mathcal{C}^{(2)}_{k-1}$
$\mathcal{R}^{(2)}_k$
$\mathcal{R}^{(3)}_k$
$\mathcal{C}^{(2)}_k$
$\mathcal{R}^{(2)}_{k+1}$
$\mathcal{R}^{(3)}_{k+1}$
$\mathcal{C}^{(2)}_{k+1}$

Fig. 3: Example of a reachability graph $\mathcal{G}_\mathcal{R}$. Nodes of the same color have connected drivable areas within one step and belong to one driving corridor (see Sec. II-C and Fig. 4).

polytopes are closed under linear maps and intersections; because of the restriction to two-dimensional polytopes, the unfavorable computational complexity of polytopes for high dimensions is irrelevant to our application. To simplify the notation, we denote the collection of $\mathcal{R}^{(i)}_k$ by $\mathcal{R}_k$, i.e., $\mathcal{R}_k = \{\mathcal{R}^{(1)}_k, \dots, \mathcal{R}^{(i)}_k\}$. The projection of $\mathcal{R}^{(i)}_k$ onto the position domain yields a axis-aligned rectangle $\mathcal{D}^{(i)}_k$ (Fig. 2c), whose union is referred to as *drivable area* $\mathcal{D}_k$. Similarly, we use $\mathcal{D}_k$ to denote the collection of $\mathcal{D}^{(i)}_k$.

We also require the reachability graph $\mathcal{G}_\mathcal{R}$, which stores the spatio-temporal relationships of the base sets $\mathcal{R}^{(i)}_k$ as a directed, acyclic graph (Fig. 3). In $\mathcal{G}_\mathcal{R}$, each node corresponds to one base set $\mathcal{R}^{(i)}_k$ and a connecting edge between two base sets $\mathcal{R}^{(i)}_k$ and $\mathcal{R}^{(j)}_{k+1}$ indicates that $\mathcal{R}^{(j)}_{k+1}$ is reachable from $\mathcal{R}^{(i)}_k$ after one step.

C. Driving Corridor

We follow the definition of *driving corridors* presented in [16]. At step k, the drivable area $\mathcal{D}_k$ may be disconnected due to the presence of obstacles, thus we introduce the notion of *connected sets* $\mathcal{C}^{(n)}_k \subseteq \mathcal{D}_k, n \in \mathbb{N}_0$ within the drivable area. A sequence of connected sets over steps 0 to k_f, where k_f is the final step, yields a driving corridor $\mathcal{C}(\cdot) = (\mathcal{C}^{(m)}_0, \dots, \mathcal{C}^{(n)}_{k_f})$. At every step, the drivable area can contain several connected sets, thus multiple driving corridors may exist. Fig. 3 shows two different driving corridors identified within a reachability graph. Therein, nodes of the same color collectively represent one driving corridor. The two driving corridors are visualized in Fig. 4. Each corridor corresponds

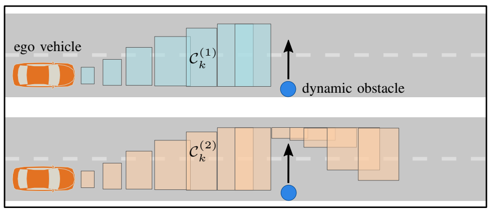


Fig. 4: Example of two driving corridors extracted from the reachability graph in Fig. 3. The driving corridors correspond to a braking maneuver (top) and an evasive maneuver (bottom). Each rectangle represents a connected set $\mathcal{C}^{(n)}_k$ at different steps $k \in \{0, \dots, k_f\}$.

to a possible maneuver of the ego vehicle with respect to the crossing obstacle.

Driving corridors can be separated into longitudinal and lateral driving corridors. This is particularly useful for motion planners that independently plan the longitudinal and lateral motions. A possible implementation is provided in [16]: Given a longitudinal driving corridor and the positions $p_{\zeta,k}$ of a planned longitudinal trajectory, a lateral driving corridor is obtained by identifying the connected sets within the longitudinal driving corridor, which contain the positions $p_{\zeta,k}$ for all $k \in \{0, \dots, k_f\}$. The search space for the lateral planning problem is thus further constrained by the fact that a lateral driving corridor in [16] is a subset of a longitudinal driving corridor.

III. Implementation Details

A. Overview

CommonRoad-Reach provides both the computation of reachable sets and the extraction of driving corridors. Subsequently, we present an overview of the toolbox and highlight its core modules (Fig. 5). The toolbox consists of the following core modules:

- `ReachableSetInterface` serves as the user interface for setting configurations, computing reachable sets and drivable areas, and extracting driving corridors.
- `ReachableSet` (Python) is an abstract superclass that instantiates the subclasses for computing reachable sets using either polytopic set propagation or graph-based propagation method.
- `PyReachableSet` is a Python implementation of the reachable set computation using the polytopic set propagation method described in [2].
- `CppReachableSet` interacts with `ReachableSet` (C++), which is a C++ implementation of the polytopic set propagation method.
- `PyGraphReachableSetOffline` precomputes reachability graphs for the graph-based reachability analysis according to [3].
- `PyGraphReachableSetOnline` loads the precomputed reachability graphs and performs the online computations of the graph-based reachability analysis.
- `DrivingCorridorExtractor` implements functions to extract possible driving corridors from a reach-

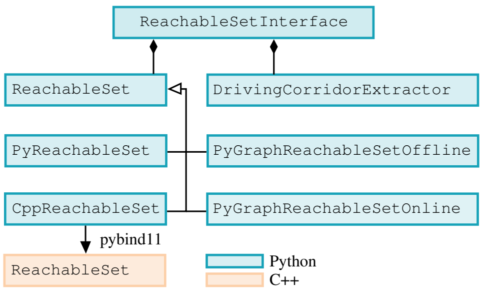


Fig. 5: UML class diagram of the core modules of our toolbox.

ability graph $\mathcal{G}_{\mathcal{R}}$.

B. Reachable Set Computation

1) Polytopic set propagation method: This method is implemented in Python and C++. In addition to [2], we support the computation in a local curvilinear coordinate system of the ego vehicle. The computation executes the following steps, as shown in Fig. 6:

1) Propagation: Each base set $\mathcal{R}^{(i)}_{k-1} \in \mathcal{R}_{k-1}$ of the previous step is propagated according to the system model (1), resulting in the propagated sets $\mathcal{R}^{\mathrm{P},(i)}_{k}$ and their corresponding drivable areas $\mathcal{D}^{\mathrm{P},(i)}_{k}$ projected onto the position domain.
2) Repartition: $\mathcal{D}^{\mathrm{P},(i)}_{k}$ are merged and repartitioned using a sweep line algorithm and a segment tree. This step helps reduce the number of rectangles and thus reduces overall computation time.
3) Collision detection: The repartitioned rectangles $\mathcal{D}^{\mathrm{P},(q)}_{k}$ are checked for collision with obstacles using the CommonRoad Drivability Checker [38]. The colliding rectangles are recursively split into two new equally sized rectangles along their longer axis. This is repeated until there is no more collision or the diagonal of the rectangle is smaller than a user-specified threshold.
4) Creation of new base sets: The new base sets $\mathcal{R}^{(j)}_{k}$ are created by determining the reachable velocities for the collision-free drivable areas $\mathcal{D}^{(j)}_{k}$.

We refer the readers to [2, Alg. 1] for more computational details.

2) Graph-based propagation method: The second provided method is the graph-based propagation method presented in [3]. In contrast to the polytopic propagation, the reachable sets are derived by traversing a precomputed reachability graph $\mathcal{G}_{\mathcal{R}}$ and removing edges that collide with the forbidden states. To use a generic, offline-computed graph for any initial state $\boldsymbol{x}_0$ and forbidden states $\mathcal{F}_k$, we precompute it assuming an initial state $\boldsymbol{x}_{0,\mathrm{off}} = \mathbf{0}$ (four-dimensional zero vector, see (1)), without considering $\mathcal{F}_k$, and by discretizing the reachable sets $\mathcal{R}_k$ using a regular grid in the position plane. We add additional edges in $\mathcal{G}_{\mathcal{R}}$ representing the reachability between sets $\mathcal{R}^{(i)}_{k}$ and $\mathcal{R}^{(j)}_{k+\kappa}$ with $\kappa \in \{1,\dots,n_{\mathrm{MS}}\}$ across up to $n_{\mathrm{MS}} \in \mathbb{N}$ steps to

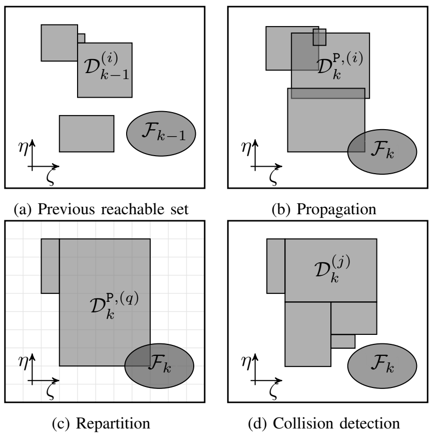


Fig. 6: Selected steps in the polytopic set propagation method. We show the reachable sets projected onto the position domain.

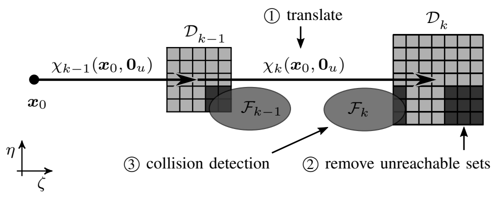


Fig. 7: Three main steps of the graph-based propagation method to remove unreachable nodes from the offline-computed graph. We show the corresponding reachable sets projected onto the position domain.

avoid the accumulation of discretization errors during the online phase. The online computation subsequently executes the following steps (Fig. 7) to account for the initial state and forbidden states:

1) Translation: To consider a given initial state $\boldsymbol{x}_0$, the offline-computed reachable set $\mathcal{R}_k$ is translated along the trajectory of the zero-input response $\chi_k(\boldsymbol{x}_0, \mathbf{0}_u)$ starting in $\boldsymbol{x}_0$.
2) Determining reachability: To determine the reachability of each set $\mathcal{R}^{(i)}_{k}$, we check which of the corresponding nodes in $\mathcal{G}_{\mathcal{R}}$ are connected for all $\kappa \in \{1,\dots,n_{\mathrm{MS}}\}$ to at least one parent node $\mathcal{R}^{(j)}_{k-\kappa}$. Nodes are removed from $\mathcal{G}_{\mathcal{R}}$ if they are not reachable. This principle is shown in Fig. 8.
3) Removal of forbidden states: Nodes whose corresponding set $\mathcal{R}^{(i)}_{k}$ intersects with $\mathcal{F}_k$ are removed from $\mathcal{G}_{\mathcal{R}}$. We perform this check by discretizing obstacles using the same road grid as in the computation of $\mathcal{R}_k$.

After the above steps, we can optionally traverse the graph backward in time, starting at the final step k_f to discard all nodes from which the final set $\mathcal{R}_{k_f}$ cannot be reached. We apply a similar principle as in step 2 and remove nodes $\mathcal{R}^{(j)}_{k-1}$ from $\mathcal{G}_{\mathcal{R}}$ without any reachable children in $\mathcal{R}_k$.

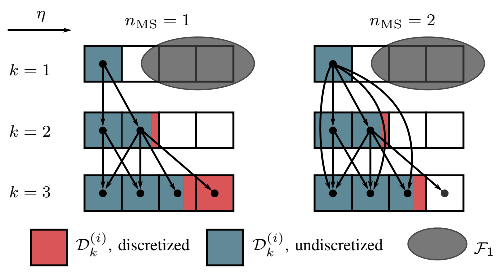


Fig. 8: One-dimensional example for reducing the discretization error of [3]: By using edges across $n_{\mathrm{MS}} > 1$ steps to determine the reachable sets, we prevent the aggregation of discretization errors in intermediate steps. The undiscretized drivable area is shown for comparison.

3) Comparison of the two methods: The polytopic set propagation method can handle time-varying bounds on velocities (2a) and accelerations (2b) of the vehicle originating from, e.g., road curvature, traffic rules, and handcrafted rules [21]. In contrast, this is not possible in the graph-based method due to the fixed parameters in the precomputation of the reachability graph. This characteristic causes greater over-approximation of the computation results in the curvilinear coordinate systems due to the previously mentioned time-varying velocity and acceleration constraints. On the other hand, the graph-based method skips geometric operations, such as splitting of reachable sets, thus the computation is often more efficient than the polytopic set propagation.

C. Driving Corridor Extraction

Our toolbox can further extract driving corridors from a reachability graph generated from Sec. III-B. To this end, we begin at the final step k_f and use a similar graph traversal procedure described at the end of Sec. III-B.2 with the following modifications: Before checking reachability between sets $\mathcal{R}_k^{(i)}$ and $\mathcal{R}_{k-1}^{(j)}$, we identify the connected sets $\mathcal{C}_k^{(n)}$ at each step k. Then, we perform the graph traversal procedure with $\kappa = 1$, i.e., we check the reachability of nodes within the connected set $\mathcal{C}_k^{(n)}$ over one step. After traversing $\mathcal{G}_\mathcal{R}$ backwards in time and identifying the connected sets $\mathcal{C}_k^{(n)}$ for all $k \in \{0, \dots, k_f\}$, the relationships between connected sets $\mathcal{C}_k^{(n)}$ and $\mathcal{C}_{k-1}^{(m)}$ of consecutive steps are stored in a separate graph $\mathcal{G}_C$ in which each path from $\mathcal{C}_0^{(m)}$ to $\mathcal{C}_{k_f}^{(n)}$ corresponds to a driving corridor (see Sec. II-C). Optionally, one can constrain driving corridors to end in a user-specified terminal set $\mathcal{I}_{k_f} \subseteq \mathbb{R}^2$ in the position domain (e.g., a set of goal states).

As an additional option, lateral driving corridors (see Sec. II-C) can be determined using the same procedure by invoking the extraction with a driving corridor and a longitudinal trajectory $(p_{\zeta,0}, \dots, p_{\zeta,k_f})$ computed by, e.g., an optimization-based motion planner. The procedure for extracting the corridor is similar to the description above, with the addition that during backwards traversal of $\mathcal{G}_\mathcal{R}$, parent sets $\mathcal{R}_{k-1}^{(j)}$ that do not contain the position $p_{\zeta,k-1}$ of the given longitudinal trajectory are excluded.

TABLE I: Parameters Used in Numerical Experiments for Different Coordinate Systems

Parameter	Unit	Cartesian	Curvilinear
k_f	step	30	30
Δ_t	s	0.1	0.1
$\overline{v}_\zeta$	$\mathrm{m/s}$	20.0	20.0
$\underline{v}_\zeta$	$\mathrm{m/s}$	-20.0	0
$\overline{v}_\eta$	$\mathrm{m/s}$	20.0	4.0
$\underline{v}_\eta$	$\mathrm{m/s}$	-20.0	-4.0
$\overline{a}_\zeta$	$\mathrm{m/s^2}$	6.0	6.0
$\underline{a}_\zeta$	$\mathrm{m/s^2}$	-6.0	-6.0
$\overline{a}_\eta$	$\mathrm{m/s^2}$	6.0	2.0
$\underline{a}_\eta$	$\mathrm{m/s^2}$	-6.0	-2.0

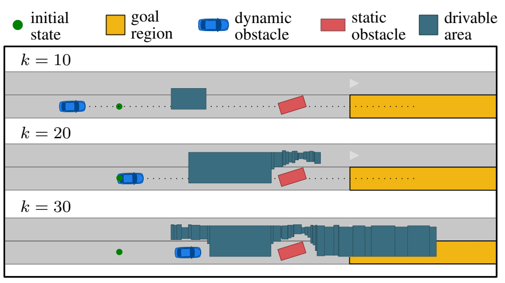


Fig. 9: Drivable areas in scenario I at different steps.

IV. Numerical Experiments

In this section, we demonstrate the key features of our toolbox by numerical experiments using three different scenarios from the CommonRoad benchmark suite with the following benchmark IDs:

I Urban road: `ZAM_Test-1_1_T-1:2020a`
II Intersection: `ARG_Carcarana-1_1_T-1:2020a`
III Highway: `USA_US101-6_1_T-1:2020a`

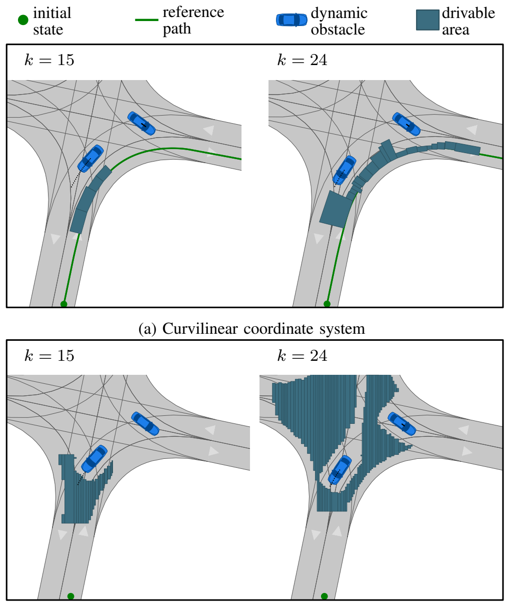
We list the main parameters used in the experiments for both coordinate systems in Tab. I. The animations of the experiments can be found at *https://mediatum.ub.tum.de/1662399*.

A. Scenario I: Urban road

The first scenario illustrates an urban driving situation with a static obstacle (e.g., a parked vehicle) in front of the ego vehicle and another vehicle following the ego vehicle. We demonstrate the computation of reachable sets for the ego vehicle in the Cartesian coordinate system. Fig. 9 shows the collision-free drivable areas for selected steps, which consider both static and time-varying occupancies of obstacles. It can be seen that the drivable areas detect the narrow passage on the left side of the static obstacle.

B. Scenario II: Intersection

Our second scenario is a four-way intersection with the presence of two other vehicles. In Fig. 10 we visualize the collision-free drivable areas at steps $k = 15$ and $k = 24$ within the Cartesian coordinate system and a curvilinear coordinate system.



(a) Curvilinear coordinate system

(b) Cartesian coordinate system

Fig. 10: Drivable areas in scenario II at different steps within two coordinate systems. We remind that the parameters used in the computations within the two coordinate systems are different (see Sec. II-A and Tab. I).

We further use this scenario to demonstrate the extraction of driving corridors (Fig. 11). To this end, we investigate the drivable area at the final step $k_f = 30$ in Fig. 11a: We see that the drivable area is disconnected due to the turning vehicle V_1 at the intersection. Thus, the drivable area exhibits two connected sets $\mathcal{C}_{30}^{(1)}$ and $\mathcal{C}_{30}^{(2)}$, each belonging to a separate driving corridor. Starting from the two connected sets in the last step, our toolbox identifies two driving corridors $\mathcal{C}_{\text{brake}}(\cdot)$ and $\mathcal{C}_{\text{turn}}(\cdot)$ for the time interval $[0, 30]$. The extracted corridors are visualized in Fig. 11b and 11c, where the corridors and the occupancy of vehicle V_1 are stacked over time. The two driving corridors correspond to different tactical decisions: In $\mathcal{C}_{\text{brake}}(\cdot)$, the ego vehicle would brake in the middle of the intersection before vehicle V_1. In contrast, corridor $\mathcal{C}_{\text{turn}}(\cdot)$ represents a maneuver where the ego vehicle would accelerate and continue turning right through the gap between V_1 and the road boundary.

C. Scenario III: Highway

Fig. 12 shows a highway scenario that is created from the NGSIM dataset [39]. We carry out the computation in a curvilinear coordinate system using a planned route as the reference path. The results show that our toolbox robustly handles the computation of the reachable sets of the ego vehicle in scenarios with multiple dynamic obstacles and can detect narrow gaps between vehicles, see, e.g., $k = 30$ in Fig. 12.

TABLE II: Computation Time Within Cartesian Frame

Method	k_f	**Unit**	**Avg.**	**Std. Dev.**
Polytopic set	30	ms	378	131
Graph-based	30	ms	227	58

TABLE III: Computation Time Within Curvilinear Frame

Method	k_f	**Unit**	**Avg.**	**Std. Dev.**
Polytopic set	30	ms	53	16
Graph-based	30	ms	26	8

D. Computation Time

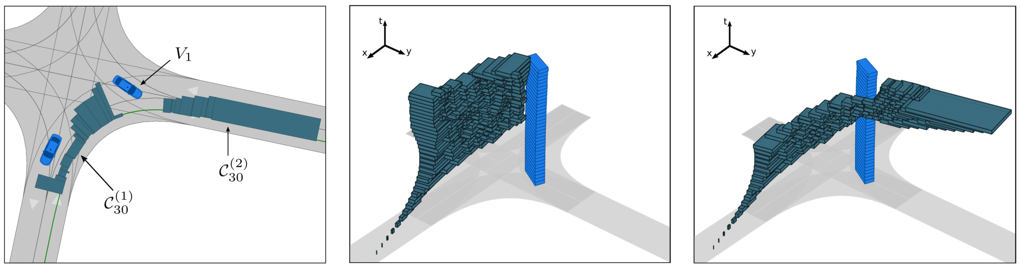
We computed the reachable sets for the planning problems given in 100 randomly chosen scenarios from the CommonRoad benchmark suite to benchmark the performance of our toolbox. All computations were executed on a laptop with an Intel Core i7-7700HQ 2.8 GHz processor. The computation times for the two methods provided in the toolbox are displayed in Tab. II for the Cartesian coordinate system and in Tab. III for curvilinear coordinate systems, respectively. For both propagation methods, computations in the curvilinear coordinate system are faster than in the Cartesian coordinate system. This is because in curvilinear coordinate systems, we used more conservative parameters in both the longitudinal and lateral directions (see Sec. II-A and Tab. I), which resulted in less nodes in the reachability graph and smaller drivable areas. As expected, the average computation times of the graph-based propagation are shorter than those of the polytopic set propagation due to the reasons described in Sec. III-B.3. The computations took a fraction of the planning horizon and render the toolbox suitable for real-time applications.

V. Conclusions

We presented CommonRoad-Reach, an open-source toolbox for computing reachable sets and extracting driving corridors for AVs. Unlike existing tools that offer general-purpose reachability algorithms, our toolbox is tailored to AV-specific applications such as motion planning in arbitrary dynamic traffic scenarios. As a result, our toolbox integrates two methods for the reachable set computation published in [2], [3]. By providing Python and C++ implementations of the algorithms, our toolbox offers both prototyping and real-time capabilities to users. From reachable sets, our toolbox further extracts collision-free driving corridors, which can be used as solution space for motion planners. We used different dynamic traffic scenarios of varied complexity to demonstrate the functionalities of our toolbox, and we benchmarked the real-time functionality against 100 scenarios from the CommonRoad benchmark suite.

Acknowledgments

This work was funded by the Huawei-TUM collaboration project *Research on Key Technologies of Safety Assurance for Autonomous Vehicles*, Deutsche Forschungsgemeinschaft (German Research Foundation) within the Priority Program



(a) Connected sets at $k_f = 30$. (b) Corridor $\mathcal{C}_{\text{brake}}(\cdot)$: Braking before V_1 (c) Corridor $\mathcal{C}_{\text{turn}}(\cdot)$: Turning right in front of V_1

Fig. 11: Identification of two driving corridors corresponding to different driving maneuvers. In figures (b) and (c) we ignore the occupancy of the other vehicle on the opposite lane for visualization purposes.

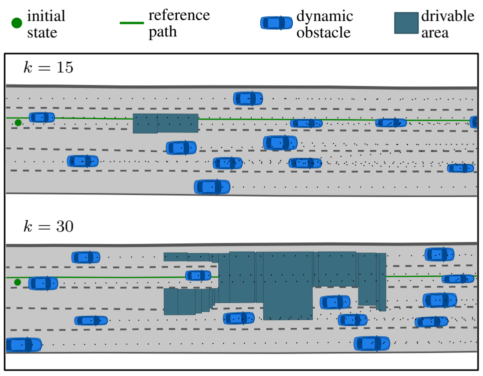


Fig. 12: Drivable areas in scenario III at different steps.

SPP 1835 *Cooperative Interacting Automobiles* under grant No. AL 1185/4-2, the German Federal Ministry of Education and Research (BMBF) within the *Munich Cluster for the Future of Mobility in Metropolitan Regions (MCube)* under grant 03ZU1105AA, and the Central Innovation Programme of the German Federal Government under grant ZF4086013GR9.

References

[1] M. Althoff, G. Frehse, and A. Girard, "Set propagation techniques for reachability analysis," *Annu. Rev. Control Rob. Auton. Syst.*, vol. 4, no. 1, pp. 369–395, 2021.

[2] S. Söntges and M. Althoff, "Computing the drivable area of autonomous road vehicles in dynamic road scenes," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 6, pp. 1855–1866, 2018.

[3] M. Klischat and M. Althoff, "A multi-step approach to accelerate the computation of reachable sets for road vehicles," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2020, pp. 2306–2312.

[4] N. Kochdumper, P. Gassert, and M. Althoff, "Verification of collision avoidance for CommonRoad traffic scenarios," in *Int. Workshop Appl. Verif. Contin. and Hybrid Syst.*, 2021, pp. 184–194.

[5] M. Althoff and J. M. Dolan, "Online verification of automated road vehicles using reachability analysis," *IEEE Trans. Rob.*, vol. 30, no. 4, pp. 903–918, 2014.

[6] S. Söntges, M. Koschi, and M. Althoff, "Worst-case analysis of the time-to-react using reachable sets," in *Proc. of the IEEE Intell. Veh. Symp.*, 2018, pp. 1891–1897.

[7] A. Lawitzky, A. Nicklas, D. Wollherr, and M. Buss, "Determining states of inevitable collision using reachability analysis," in *Proc. of the IEEE Int. Conf. Intell. Robot. Syst.*, 2014, pp. 4142–4147.

[8] M. Koschi and M. Althoff, "Set-based prediction of traffic participants considering occlusions and traffic rules," *IEEE Trans. Intell. Veh.*, vol. 6, no. 2, pp. 249–265, 2020.

[9] M. Naumann, H. Konigshof, M. Lauer, and C. Stiller, "Safe but not overcautious motion planning under occlusions and limited sensor range," in *Proc. of the IEEE Intell. Veh. Symp.*, 2019, pp. 140–145.

[10] P. F. Orzechowski, A. Meyer, and M. Lauer, "Tackling occlusions and limited sensor range with set-based safety verification," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2018, pp. 1729–1736.

[11] T. Stahl and F. Diermeyer, "Online verification enabling approval of driving functions – Implementation for a planner of an autonomous race vehicle," *IEEE Open J. Intell. Transp. Syst.*, vol. 2, pp. 97–110, 2021.

[12] Y. S. Shao, C. Chen, S. Kousik, and R. Vasudevan, "Reachability-based trajectory safeguard: A safe and fast reinforcement learning safety layer for continuous control," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 3663–3670, 2021.

[13] C. Pek and M. Althoff, "Fail-safe motion planning for online verification of autonomous vehicles using convex optimization," *IEEE Trans. Rob.*, vol. 37, no. 3, pp. 798–814, 2020.

[14] H. Krasowski, X. Wang, and M. Althoff, "Safe reinforcement learning for autonomous lane changing using set-based prediction," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2020, pp. 1–7.

[15] S. Vaskov, H. Larson, S. Kousik, M. Johnson-Roberson, and R. Vasudevan, "Not-at-fault driving in traffic: A reachability-based approach," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2019, pp. 2785–2790.

[16] S. Manzinger, C. Pek, and M. Althoff, "Using reachable sets for trajectory planning of automated vehicles," *IEEE Trans. Intell. Veh.*, vol. 6, no. 2, pp. 232–248, 2021.

[17] S. Söntges and M. Althoff, "Computing possible driving corridors for automated vehicles," in *Proc. of the IEEE Intell. Veh. Symp.*, 2017, pp. 160–166.

[18] T. Gu, J. M. Dolan, and J.-W. Lee, "Automated tactical maneuver discovery, reasoning and trajectory planning for autonomous driving," in *Proc. of the IEEE Int. Conf. Intell. Robot. Syst.*, 2016, pp. 5474–5480.

[19] P. Bender, O. S. Tas, J. Ziegler, and C. Stiller, "The combinatorial aspect of motion planning: Maneuver variants in structured environments," in *Proc. of the IEEE Intell. Veh. Symp.*, 2015, pp. 1386–1392.

[20] G. Würsching and M. Althoff, "Sampling-based optimal trajectory generation for autonomous vehicles using reachable sets," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2021, pp. 828–835.

[21] E. Irani Liu and M. Althoff, "Computing specification-compliant reachable sets for motion planning of automated vehicles," in *Proc. of the IEEE Intell. Veh. Symp.*, 2021, pp. 1037–1044.

[22] S. Manzinger and M. Althoff, "Tactical decision making for cooperative vehicles using reachable sets," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2018, pp. 444–451.

[23] X. Chen, S. Sankaranarayanan, and E. Abraham, "Flow* 1.2: More effective to play with hybrid systems," in *Int. Workshop Appl. Verif. Contin. and Hybrid Syst.*, 2015, pp. 152–159.

[24] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "SpaceEx: Scalable verification of hybrid systems," in *Proc. of the Int. Conf. Comput. Aided Verif.*, 2011, pp. 379–395.

[25] M. Althoff, "An introduction to CORA 2015," in *Proc. of the Workshop Appl. Verif. Contin. and Hybrid Syst.*, 2015, pp. 120–151.

[26] S. Bogomolov, M. Forets, G. Frehse, F. Viry, A. Podelski, and C. Schilling, "Reach set approximation through decomposition with low-dimensional sets and high-dimensional matrices," in *Proc. of the Int. Conf. Hybrid Syst.: Comput. and Control*, 2018, pp. 41–50.

[27] S. Bak and P. S. Duggirala, "HyLAA: A tool for computing simulation-equivalent reachability for linear systems," in *Proc. of the Int. Conf. Hybrid Syst.: Comput. and Control*, 2017, pp. 173–178.

[28] P. S. Duggirala, S. Mitra, M. Viswanathan, and M. Potok, "C2E2: A verification tool for stateflow models," in *Int. Conf. Tools Algorithms Const. Anal. Syst.*, 2015, pp. 68–82.

[29] I. Xausa, R. Baier, O. Bokanowski, and M. Gerdts, "Computation of avoidance regions for driver assistance systems by using a Hamilton-Jacobi approach," *Optim. Control. Appl. Methods*, vol. 41, no. 2, pp. 668–689, 2020.

[30] J. F. Fisac, M. Chen, C. J. Tomlin, and S. S. Sastry, "Reach-avoid problems with time-varying dynamics, targets and constraints," in *Proc. of the Int. Conf. Hybrid Syst.: Comput. and Control*, 2015, pp. 11–20.

[31] K. Margellos and J. Lygeros, "Hamilton-Jacobi formulation for reach-avoid problems with an application to air traffic management," in *Proc. of the Am. Control Conf.*, 2010, pp. 3045–3050.

[32] O. Bokanowski, N. Forcadel, and H. Zidani, "Reachability and minimal times for state constrained nonlinear problems without any controllability assumption," *SIAM J. Control Optim.*, vol. 48, no. 7, pp. 4292–4316, 2010.

[33] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *Proc. of the IEEE Intell. Veh. Symp.*, 2017, pp. 719–726.

[34] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *Proc. of the IEEE Intell. Veh. Symp.*, 2014, pp. 420–425.

[35] B. Schürmann, D. Heß, J. Eilbrecht, O. Stursberg, F. Köster, and M. Althoff, "Ensuring drivability of planned motions using formal methods," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2017, pp. 1–8.

[36] M. Althoff and J. M. Dolan, "Reachability computation of low-order models for the safety verification of high-order road vehicle models," in *Proc. of the Am. Control Conf.*, 2012, pp. 3559–3566.

[37] J. Eilbrecht and O. Stursberg, "Challenges of trajectory planning with integrator models on curved roads," in *Proc. of the IFAC World Congr.*, 2020, pp. 15 588–15 595.

[38] C. Pek, V. Rusinov, S. Manzinger, M. C. Üste, and M. Althoff, "CommonRoad Drivability Checker: Simplifying the development and validation of motion planning algorithms," in *Proc. of the IEEE Intell. Veh. Symp.*, 2020, pp. 1013–1020.

[39] V. Alexiadis, J. Colyar, J. Halkias, R. Hranac, and G. McHale, "The next generation simulation program," *Inst. Transp. Eng. ITE J.*, vol. 74, no. 8, p. 22, 2004.

A.6 Scenario Factory: Creating Safety-Critical Traffic Scenarios for Automated Vehicles [6]

Summary Testing, verifying, and comparing the safety of automated vehicles' motion planning algorithms necessitates a significant amount of simulated traffic scenarios. This work proposes a novel method to automatically generate a large number of safety-critical traffic scenarios, which can be used for testing various applications of automated vehicles.

We start by extracting a large number of road networks across the globe from OpenStreetMap (this functionality is referred to as *Globetrotter*). Among these extracted road networks, we identify particularly interesting ones to create distinct scenarios with diverse characteristics. Following this, we utilize the SUMO traffic simulator to populate the road networks with traffic participants, thereby generating many primitive scenarios. Lastly, we create challenging scenarios for motion planning of automated vehicles by increasing the criticality of the primitive scenarios using reachability analysis and nonlinear optimization techniques.

In contrast to existing works on generating test scenarios using simulation, our approach is very efficient since we automate the extraction of road networks and the simulation of traffic participants. By leveraging the extensive amount of road networks across the globe, we are able to create complex yet realistic road networks that currently no procedural map generator is capable of producing. Moreover, unlike previous methods, our approach does not depend on data obtained through on-road test drives or other real-world traffic sources. As a result, our approach allows for the efficient creation of a large number of traffic scenarios at a meager cost. Furthermore, our scenarios have the property that they are independent of the vehicle under test since our criticality measure is defined based on the amount of so-called *drivable area*. This provides versatile applicability to a wide range of automated vehicles with diverse vehicle models. We offer the generated scenarios to the public via the CommonRoad website.

Contributions of E. I. L. E. I. L. supervised the bachelor thesis of F. H., which shaped the main body of Globetrotter; E. I. L. designed, conducted, and evaluated the experiments (together with M. K.); E. I. L. wrote the article (together with M. K., F. H., and M. A.).

Conference article The accepted version of the article is reprinted. The final version of record is available at `https://doi.org/10.1109/ITSC45102.2020.9294629`.

Scenario Factory: Creating Safety-Critical Traffic Scenarios for Automated Vehicles

Moritz Klischat*, Edmond Irani Liu*, Fabian Höltke, and Matthias Althoff

***Abstract*— The safety validation of motion planning algorithms for automated vehicles requires a large amount of data for virtual testing. Currently, this data is often collected through real test drives, which is expensive and inefficient, given that only a minority of traffic scenarios pose challenges to motion planners. We present a workflow for generating a database of challenging and safety-critical test scenarios that is not dependent on recorded data. First, we extract a large variety of road networks across the globe from OpenStreetMap. Subsequently, we generate traffic scenarios for these road networks using the traffic simulator SUMO. In the last step, we increase the criticality of these scenarios using nonlinear optimization. Our generated scenarios are publicly available on the CommonRoad website.**

I. Introduction

Virtual testing is an important tool for validating the safety of automated vehicles, as it exposes potential defects of the algorithms under test. Having a large variety of challenging traffic scenarios is vital for effective and efficient testing of motion planning algorithms. While carrying out simulations using data recorded from test drives provides us with realistic scenarios, the required data collection is often overly expensive and time-consuming [1]. Even though the number of publicly-available datasets has increased over the last few years, e.g., [2]–[5], they usually feature only a small number of maps and require much effort to record.

Our framework generates a database of safety-critical scenarios for scenario-based testing [6] of motion planing algorithms for automated vehicles. It consists of

1) Extracting interesting road intersections worldwide from OpenStreetMap (OSM) [7] by using our Globetrotter tool (see Sec. III).
2) Generating safety-critical test scenarios by first populating the extracted intersections with traffic participants through the traffic simulator SUMO [8].
3) Optimizing the criticality of the obtained scenarios by using a generalizable criticality criterion (see Sec. IV).

A. Related Work

Below, we concisely review related works on approaches towards automatically creating virtual representations of road networks and generating critical test scenarios for automated vehicles.

*The first two authors have contributed equally to this work.

All authors are with the Department of Informatics, Technical University of Munich, 85748 Garching, Germany.
{moritz.klischat, edmond.irani, fabian.hoeltke, althoff}@tum.de

1) Creating road networks: Generative approaches construct road networks, e.g., based on abstract specifications [9]. Moreover, a suite of road networks with a defined coverage of road curvatures is generated in [10] using satisfiability modulo theories. Road networks that lead to the failure of lane-keeping assistants are generated procedurally in [11] through mutating road networks using genetic algorithms.

Alternatively, road networks can also be created from external sources. In [12]–[16], the authors extract road networks from aerial and satellite images with the help of computer vision techniques. These works are capable of extracting high-level geometric information of road networks; however, lane-level information concerning motion planners of automated vehicles is not reconstructed. Promising works towards the reconstruction of road networks with lane-level detail from aerial images can be seen in [17], [18]. Similarly, while creating road networks for single lanes from OSM data is straightforward [19], creating those with lane-level detail is a more challenging problem [20].

2) Generating critical test scenarios: Creating test scenarios for automated vehicles using traffic simulators is proposed, e.g., in [21], [22]. Realistic scenarios can be obtained by calibrating their simulations through real-world measurements. By using criticality metrics, safety-critical scenarios are filtered [22]; however, these situations occur only rarely, in the main.

To efficiently obtain critical scenarios, importance sampling from large databases of recorded traffic data is proposed [23]. Criticality metrics are combined with the occurrence rates to efficiently sample critical scenarios representative of real-world driving conditions [24]. Other approaches use optimization to create critical scenarios based on these metrics [25], [26]. Similarly, falsification methods can detect scenarios that falsify a motion planner with respect to a given safety specification [27], [28]. Parameter regions for critical scenarios based on constraint satisfaction are computed in [29]. In our previous work, we presented an optimization-based method to increase the criticality of initially uncritical traffic scenarios [30], [31] by decreasing the space of possible solutions for the vehicle under test, called the *drivable area*.

B. Contributions

In contrast to previous work on generating test scenarios through simulation, our approach is particularly efficient since we combine the automatic extraction of road networks from OSM with a traffic simulation followed by an increase of its criticality. By exploiting the large variety of road networks around the world, we create complex, yet realistic road networks which currently no procedural map generator is capable of producing. In contrast to existing work, our approach does not rely on test drives nor other real-world traffic data, thus it is able to efficiently create many traffic scenarios at low costs. The resulting scenarios are independent of the vehicle under test, due to our criticality metric based on the drivable area.

II. Overview

An overview of our approach is presented in Fig. 1. In the following subsections, we introduce each component.

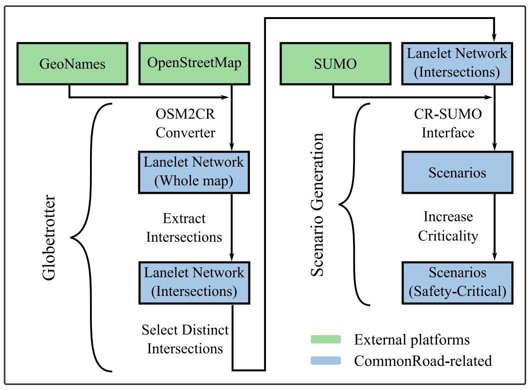


Fig. 1. Our pipeline for generating safety-critical scenarios.

A. Platforms

1) CommonRoad: The CommonRoad (CR) benchmark suite[1] is an open-source framework that provides a collection of traffic scenarios for motion planning algorithms. Scenarios in CommonRoad consist of *road networks*, *static obstacles*, and *dynamic obstacles* that represent all possible types of traffic participants. In this work, we focus on cars, trucks, and bicycles. *Road networks* in CommonRoad are described by lanelets [32] (see Fig. 2). Lanelets are defined by their left and right bounds, which are modeled by polylines. Furthermore, lanelets are connected through successor-predecessor and lateral-adjacency relations and contain additional information such as the speed limit. Additionally, we define *forking points* as the points on the centerlines of lanelets where lanelets split or multiple lanelets merge.

2) OpenStreetMap: OSM is an open-source project that provides geographic data worldwide. The main structure of OSM data is defined by three elements: *nodes*, *ways*, and *relations*. *Nodes* are geographic points defined by their latitude and longitude; *ways* are tuples of *nodes* and represent elements such as roads and boundaries of areas; *relations* are groups of *nodes*, *ways* and other *relations*. Fig. 3a shows a map taken from OpenStreetMap.

[1]https://commonroad.in.tum.de/

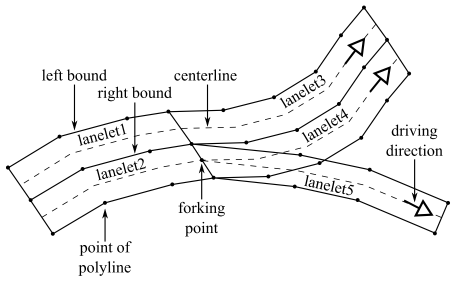


Fig. 2. Lanelet network representation.

3) GeoNames: GeoNames[2] is a free geographic database which covers all countries and contains over 11 million placenames of cities from all over the world. The provided geographical information includes global coordinates, postal codes, population, etc.

4) SUMO: This open-source microscopic traffic-simulation package is designed to handle large road networks. SUMO models individual vehicles and their interactions using models for car-following, lane-changing, and intersection behavior.

B. Converters

Since most of the software platforms mentioned above have individual formats and map representations, we use different converters and interfaces to bridge these platforms.

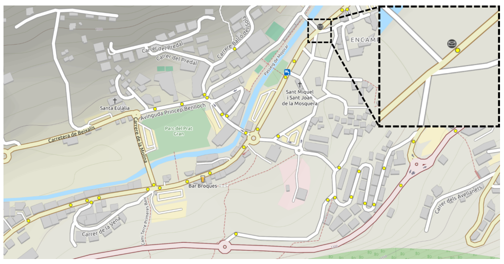
1) OSM2CR converter: It converts OSM maps to CommonRoad lanelet networks. While OSM provides map data for almost any place in the world, their level of detail is not yet suited for automated vehicles: The motion planners of automated vehicles and traffic simulators typically require lane-level information. To resolve this issue, in the first step, the topology of the lanelet network, i.e., the connections at intersections, needs to be estimated. Next, spatial information of individual lanes is deducted accordingly. Fig. 3b shows a converted lanelet network via this converter.

2) CR-SUMO interface: It enables the communication between CommonRoad and SUMO by a) converting the CR road network to SUMO format, b) generating configuration files for the simulation, and c) converting simulated vehicle trajectories to the CR format. We refer the interested reader to [33] for more details regarding this interface.

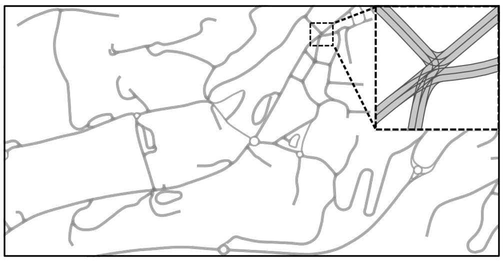
III. Globetrotter

To automatically extract interesting road networks from all over the world, we have developed the Globetrotter tool, which takes the road network data from OSM as its underlying input. As we want to create scenarios on distinct road networks, we mainly focus on extracting intersections. Below, we explain the major steps for extracting the intersections from OSM.

[2]https://www.geonames.org/



(a)



(b)

Fig. 3. (a) Map of Encamp, Andorra taken from OSM. (b) Conversion result into CommonRoad lanelet network via the OSM2CR converter.

A. Retrieving Candidate Regions

Clearly, there are intersections all over the globe, and they mostly vary according to region. Given that only 29% of the Earth's surface is covered by land[3], and that 10% of these regions accommodate 95% of the human population[4], sampling the Earth's surface with random coordinates is not very efficient. Assuming that intersections mostly occur near populated areas, we retrieve these populated candidate regions from GeoNames. To speed up the processing in the next steps, we can also divide the region into smaller subregions if the area of the region exceeds a certain value. The retrieved candidate (sub)regions are converted into lanelet networks via the OSM2CR converter.

B. Extracting Intersections

We denote the n-th forking point and the tuple of all forking points in a lanelet network as P_n and $\mathcal{P}$, respectively. For a given lanelet network, it is usually difficult to determine beforehand the number of intersections to be extracted. For this reason, instead of k-means-like algorithms [34], we apply the hierarchical agglomerative clustering (HAC) algorithm [35] to $\mathcal{P}$. HAC only requires a distance threshold d_{th} to limit the distances between clusters: a higher d_{th} entails larger intersections. Alg. 1 describes how the intersections are extracted from $\mathcal{P}$.

1) Clustering forking points (Alg. 1, lines 2-4): Initially, each forking point forms a cluster C_n with it being the only member: $C_n = \{P_n\}$. We denote the tuple of clusters by $\mathcal{C} := \langle C_1, C_2, \dots \rangle$, and the distance between two clusters C_i, C_j with single-linkage setting [35] by $d_{i,j}$:

$$d_{i,j} = \min\{\text{dist}(a,b) | a \in C_i, b \in C_j\},$$

where the operator $\text{dist}(\cdot)$ returns the Euclidean distance between two given forking points. In each iteration, the two clusters C_i, C_j with the minimum distance $d_{i,j} < d_{\text{th}}$ are merged into a new cluster $C' = C_i \cup C_j$. This process is repeated until no more clusters can be merged. Fig. 4a-4b show the dendrogram for clustering an exemplary lanelet network and the clustered forking points.

Algorithm 1 Extracting Intersections

```
    Inputs: forking points $\mathcal{P}$, distance threshold $d_{\text{th}}$
    Output: extracted intersections $\mathcal{I}$
 1: $\mathcal{I} \leftarrow \emptyset$
 2: $\triangleright$ Clustering forking points
 3: $\mathcal{C} \leftarrow$ INITIALIZE($\mathcal{P}$)
 4: $\mathcal{C} \leftarrow$ HAC($\mathcal{C}$, $d_{\text{th}}$)
 5: $\triangleright$ Creating intersections
 6: for $C' \in \mathcal{C}$ do
 7:     $I \leftarrow$ CUTLANELETS($C'$)
 8:     $I \leftarrow$ POSTPROCESS($I$)
 9:     $\mathcal{I} \leftarrow \mathcal{I} \cup \{I\}$
10: end for
11: return $\mathcal{I}$
```

2) Creating intersections (Alg. 1, lines 5-10): For each remaining cluster $C' \in \mathcal{C}$, we determine the minimum radius $r_{\min}$ of a circle enclosing all forking points within the cluster. We enlarge this radius by a user-defined margin r_{mgn} to span a region of interest. We cut out all lanelets from this region, resulting in an intersection I, and additionally apply the following steps:

1) Lanelets that are not within the region of interest are removed.
2) Due to removed lanelets, we update the successor-predecessor and lateral-adjacency relations.

Fig. 4c shows the extracted intersections $\mathcal{I}$.

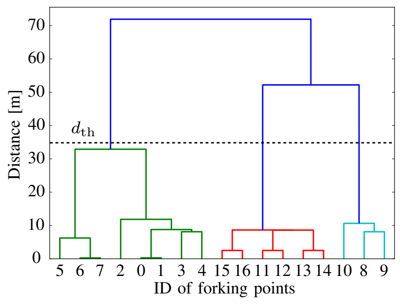
C. Selecting Interesting Intersections

Given the intersections $\mathcal{I}$ extracted from a lanelet network, we only keep those that are particularly interesting or distinct according to the following features:

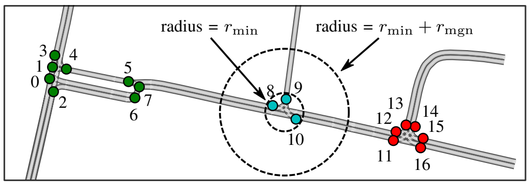
- number of forking points;
- number of lanelets;
- number of crossing lanelets;
- number of predecessors and successors;
- area of lanelets;
- density of lanelets;
- angle between lanelets; and
- mean distance between forking points and their centroid.

We associate the interestingness of intersections with the dissimilarity between their features and those of other

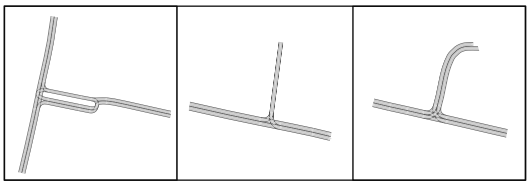
[3]https://www.noaa.gov/
[4]https://ec.europa.eu/jrc/en



(a)



(b)



(c)

Fig. 4. (a) Dendrogram of the clustering result. Three clusters are generated with d_{th} set to 35 meters. (b) Forking points within one cluster have the same color. r_{mgn} is set to 15 meters. (c) Intersections extracted from the input lanelet network.

intersections. By doing so, we turn the selection of interesting intersections into a multivariate outlier (anomaly) detection problem. To solve this problem, we use the isolation forest (iForest) algorithm [36], since it is unsupervised, capable of efficiently handling multiple dimensions of features, and requires limited effort to hand-tune its parameters. In the training phase, a total of k isolation trees (iTrees) are trained with sets of randomly-selected intersections; in the detection phase, an anomaly score $s \in [0,1]$ is assigned to each intersection by the iTrees [36], where an intersection with a score above a threshold s_{th} is considered an outlier. Fig. 5 presents a collection of distinct intersections. It should be recalled that we divide the candidate region into subregions if it is overly large, thus rendering the adopted iForest algorithm computationally tractable.

IV. Generation of Safety-Critical Scenarios

On the extracted maps, we simulate traffic participants using our previously-introduced CR-SUMO interface next. Since scenarios simulated with SUMO often yield uncritical, easy-to-solve motion planning problems, we subsequently increase their criticality using our approach [30], [31] for reducing the solution space. We first parametrize the initially obtained trajectories of other traffic participants in Sec. IV-B and, following that, formulate a nonlinear optimization problem with a criticality criterion specified in Sec. IV-D.

A. Motion Planning Problem

The system dynamics of the ego vehicle is defined by

$$\dot{x}_{\text{e}}(t) = f(x_{\text{e}}(t), u(t)),$$

where $x_{\text{e}}(t) \in \mathbb{R}^n$ is the state vector and $u(t) \in \mathcal{U}$ is the input vector with the set of admissible inputs $\mathcal{U} \in \mathbb{R}^m$.

The trajectories of n_{tp} other traffic participants are given by $x_i(t;p)$, $i \in \{1,...,n_{\text{tp}}\}$ with parameters $p \in \mathbb{R}^{n_{\text{p}}}$, initial time t_0, and final time t_f. Initial candidates for these trajectories are obtained from SUMO; their parametrization is explained in more detail in Sec. IV-B. The occupied space $\mathcal{O}_i(t;p) \subset \mathbb{R}^2$ of a traffic participant is obtained through the $\text{occ}(\cdot)$ operator, i.e., $\mathcal{O}_i(t;p) = \text{occ}(x_i(t;p))$. We define the motion planning problem for the ego vehicle as a classical reach-avoid problem: given an initial state $x_{\text{e},0} = x_{\text{e}}(t_0)$, an input trajectory $u(t)$ has to be found to steer the ego vehicle into a goal region while not leaving the road surface $\mathcal{W}_{\text{lanes}} \in \mathbb{R}^2$ and avoiding the space $\mathcal{O}(t;p)$ occupied by all obstacles, i.e.,

$$\forall t \in [t_0, t_f] : \text{occ}(x_{\text{e}}(t)) \subseteq \mathcal{W}_{\text{lanes}} \setminus \mathcal{O}(t;p). \quad (1)$$

We obtain a motion planning problem by deleting a selected vehicle in a scenario simulated with SUMO and storing the initial state $x_{\text{e},0}$ of this vehicle. Vehicles with interesting maneuvers are automatically selected by using thresholds on the velocity and acceleration profiles or by identifying lane changes, turns or vehicles driving nearby.

B. Scenario Parametrization

In order to optimize the criticality of scenarios, we parametrize trajectories by the parameter vector p. We describe trajectories in lane-based coordinate systems, in which a state is defined as $x = [s_\xi, \dot{s}_\xi, s_\eta, \dot{s}_\eta]^T$. The subscripts ξ and η denote the longitudinal and lateral coordinates with respect to the centerline, respectively (see Fig. 2).

For the n_{tp} traffic participants, we only parametrize the longitudinal trajectory using translations $p^{\text{s}} \in \mathbb{R}^{n_{\text{tp}}}$, initial velocity variations $p^{\text{v}} \in \mathbb{R}^{n_{\text{tp}}}$, and acceleration variations $p^{\text{a}} \in \mathbb{R}^{n_{\text{tp}}}$, yielding $p = [p^{\text{s}}, p^{\text{v}}, p^{\text{a}}]^T$. The parametrized longitudinal position trajectory is given by

$$s_{\xi,i}(t;p_i) = \hat{s}_{\xi,i}(t) + p_i^{\text{s}} + p_i^{\text{v}} t + \frac{1}{2} p_i^{\text{a}} t^2. \quad (2)$$

From (2), the centerlines, and the dimensions of the vehicle, we obtain the occupied space $\mathcal{O}_i(t,p)$ of each traffic participant.

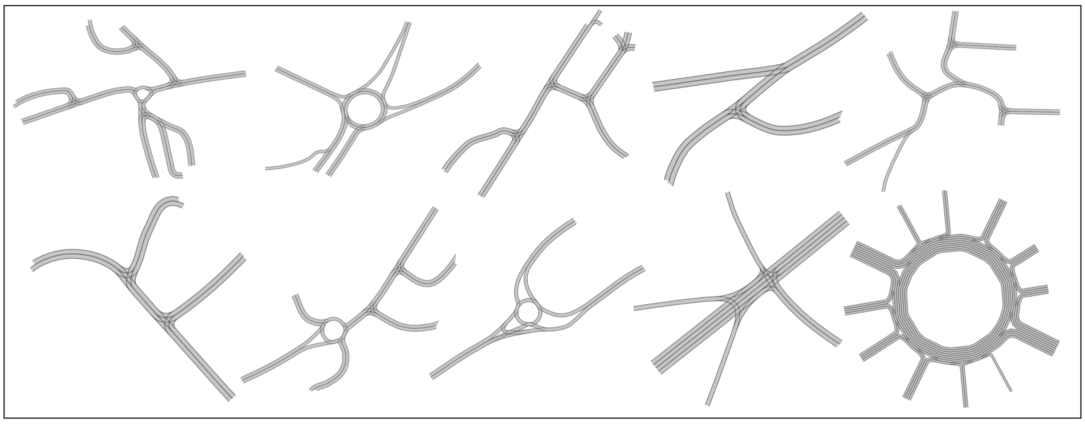


Fig. 5. Selected road intersections generated by Globetrotter ($s_{\text{th}} = 0.9$).

C. Drivable Area

We denote a feasible solution to the motion planning problem defined in Sec. IV-A as $\chi(t; x_0, u(\cdot))$, where $u(\cdot)$ refers to the entire trajectory instead of a particular value $u(t)$ at time t. To quantify the criticality of a scenario, we use the solution space, which corresponds to the set of reachable states for $t \in [t_0, t_f]$ without collisions:

$$\mathcal{R}(t; x_0, \mathcal{O}(\,\cdot\,; p)) = \Big\{ \chi(t; x_0, u(\cdot)) \,\Big|\, \forall \tau \in [t_0, t_f] : u(\tau) \in \mathcal{U}, \; \text{occ}\big(\chi(\tau; x_0, u(\cdot))\big) \subseteq \mathcal{W}_{\text{lanes}} \backslash \mathcal{O}(\tau; p) \Big\}.$$

By applying the projection operator $\text{proj}(x) : \mathbb{R}^n \to \mathbb{R}^2$, which projects the state space to the position domain, we obtain the *drivable area*

$$\mathcal{D}(t; x_0, \mathcal{O}(\,\cdot\,; p)) = \bigcup_{x \in \mathcal{R}(t; x_0, \mathcal{O}(\,\cdot\,; p))} \text{proj}(x). \tag{3}$$

An example for the drivable area in presence of an obstacle is depicted in Fig. 6.

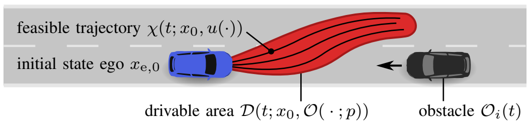

feasible trajectory $\chi(t; x_0, u(\cdot))$
initial state ego $x_{\text{e},0}$
drivable area $\mathcal{D}(t; x_0, \mathcal{O}(\,\cdot\,; p))$
obstacle $\mathcal{O}_i(t)$

Fig. 6. Example of a drivable area for the time interval $[t_0, t_f]$.

To quantify the solution space, we introduce the function $\text{area}(\mathcal{X})$ returning the area of a set. We write

$$A(t; p) := \text{area}\big(\mathcal{D}(t; x_0, \mathcal{O}(\,\cdot\,; p))\big)$$

to obtain the area profile of the drivable area over time. We compute the drivable area using our approach as in [37].

D. Optimization Problem

For increasing the criticality of the motion planning problem, we optimize the parameter vector p to obtain a desired, critical area profile $A_{\text{crit}}(t)$:

$$\underset{p}{\text{argmin}}\, \kappa(p), \quad \kappa(p) = \int_0^{t_f} \big(A(t; p) - A_{\text{crit}}(t)\big)^2 \tag{4}$$

$$\text{subject to} \qquad \forall t, \forall i, \forall j \neq i : \quad \mathcal{O}_i(t; p) \cap \mathcal{O}_j(t; p) = \emptyset. \tag{5}$$

The constraint in (5) ensures that no traffic participants collide with each other. In this work, we use the drivable area computed without any traffic participants and the scalar $\gamma \in \,]0, 1[$ which quantifies the reduction of the drivable area: $A_{\text{crit}}(t) = \gamma \cdot \text{area}(\mathcal{D}(t; x_0, \emptyset))$.

Since the drivable area is highly nonlinear with respect to the trajectories of other traffic participants and possibly subjected to local minima, we use particle swarm optimization [38] as in our previous work [30]. Furthermore, we implement a repair algorithm that enforces the collision constraint (5). To that end, we formulate the collision constraints as linear inequality constraints and correct infeasible solutions by computing the closest feasible solution using linear programming. A more efficient optimization is ensured by an a priori computation of relevant parameter intervals as presented in [30].

V. Evaluation

We demonstrate our approach by generating scenarios on a large variety of road networks from various places across the world. First, we obtain 576 road networks from 8 countries and 46 cities from Globetrotter, for each of which we simulate multiple scenarios using our CR-SUMO interface. After selecting interesting ego vehicles, we obtain 1402 scenarios for which we optimize the criticality. The resulting scenarios are added to our website[5].

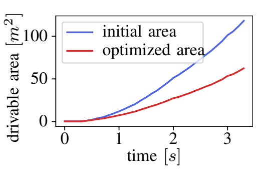
[5]https://commonroad.in.tum.de/

In Fig. 7a we compare the area profiles $A(t;p)$ of the drivable area in the optimized scenarios against the initial scenario obtained from SUMO: our approach is able to significantly decrease the drivable area, and it thus increases the criticality. Fig. 7b shows the distribution of the achieved reduction of the critical area. For most of the optimized scenarios, the drivable area ranges between 0.2 – 0.3 of its initial size.

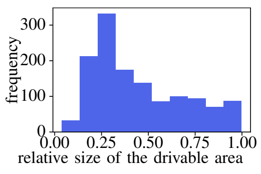
TABLE I

PARAMETERS FOR CRITICALITY OPTIMIZATION

Drivable area computation	
max. acceleration ego vehicle $\lvert a_{max} \rvert$	$5.0\,\mathrm{m/s^2}$
time step size Δt	$0.1\,\mathrm{s}$
time horizon t_f	$3.4\,\mathrm{s}$
Constraints for optimization	
initial velocity variation	$[-3,3]\,\mathrm{m/s}$
acceleration variation	$[-5,2]\,\mathrm{m/s^2}$



(a)



(b)

Fig. 7. (a) Size of the drivable area $A(t;p)$ over time, averaged over all scenarios. (b) Histogram of the size of the drivable area in the optimized scenarios relative to the initial scenarios.

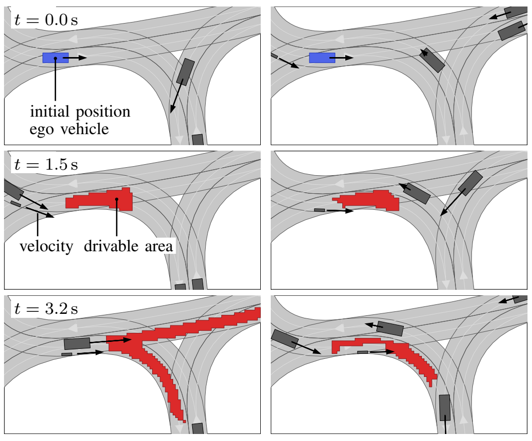
Let us present some concrete examples for demonstrating our algorithm. The first example is an intersection from the town Pula, Croatia. In Fig. 8, we compare the drivable area of the initial scenario obtained from SUMO with the optimized scenario. Note that we restrict the allowed road surface $\mathcal{W}_{\text{lanes}}$ to lanelets that the ego vehicle is allowed to drive in. In the initial scenario, the ego vehicle could either turn freely to the right or drive straight. However, after the optimization, two turning vehicles and a bicycle restrict possible maneuvers of the ego vehicle.

The second example is a four-way intersection from the town Putte, Belgium. In the optimized scenario, the ego vehicle must either respect an oncoming vehicle when turning left or a bicycle when driving straight. As a result, the drivable area is split into two parts, as shown in Fig. 9.

VI. Conclusions

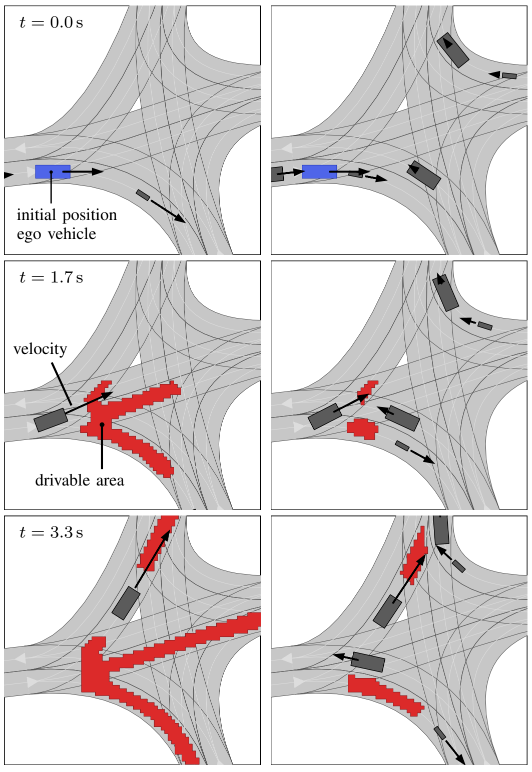
We present an approach to automatically generate a large number of test scenarios for automated vehicles. Our results show that we are able to extract a large number of distinct road networks from OpenStreetMaps, for which we simulate traffic scenarios using the traffic simulator SUMO. Our approach subsequently yields challenging scenarios by decreasing the solution space for motion planning algorithms.

The generated, publicly-available scenarios render the virtual testing of motion-planning algorithms in challenging situations easier. In the future, the explicit consideration of traffic rules during the generation of our critical scenarios will further improve our test cases.



(a) Initial scenario from simulation. (b) Optimized, more critical scenario.

Fig. 8. Example 1: Comparison of the drivable areas at different times.



(a) Initial scenario from simulation. (b) Optimized, more critical scenario.

Fig. 9. Example 2: Comparison of the drivable areas at different times.

ACKNOWLEDGMENTS

The authors would like to thank Chuxuan Li for her work on the SUMO interface and Maximilian Rieger for his work on the OSM2CR converter. We gratefully acknowledge the financial support from the Central Innovation Programme of the German Federal Government under grant no. ZF4086007BZ8 and the German Research Foundation (DFG) within the Priority Programme SPP 1835 *Cooperative Interacting Automobiles* under grant no. AL 1185/4-2.

REFERENCES

[1] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transp. Res. Part A: Policy Pract.*, vol. 94, pp. 182–193, 2016.

[2] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kümmerle, H. Königshof, C. Stiller, A. de La Fortelle, and M. Tomizuka, "INTERACTION dataset: An INTERnational, Adversarial and Cooperative moTION dataset in interactive driving scenarios with semantic maps," *arXiv:1910.03088*, 2019.

[3] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet, "Lyft Level 5 AV dataset 2019," https://level5.lyft.com/dataset/, 2019.

[4] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo Open dataset," *arXiv:1912.04838*, 2019.

[5] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highD dataset: A drone dataset of naturalistic vehicle trajectories on German highways for validation of highly automated driving systems," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2018, pp. 2118–2125.

[6] S. Riedmaier, T. Ponn, D. Ludwig, B. Schick, and F. Diermeyer, "Survey on scenario-based safety assessment of automated vehicles," *IEEE Access*, vol. 8, pp. 87 456–87 477, 2020.

[7] M. Haklay and P. Weber, "OpenStreetMap: User-generated street maps," *IEEE Pervasive Comput.*, vol. 7, no. 4, pp. 12–18, 2008.

[8] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO–simulation of urban mobility: an overview," in *Proc. of Int. Conf. Adv. Syst. Simul.*, 2011, pp. 63–68.

[9] C. Campos, J. M. Leitão, J. P. Pereira, A. Ribas, and A. F. Coelho, "Procedural generation of topologic road networks for driving simulation," in *Iberian Conf. Inf. Syst. Technol.*, 2015, pp. 1–6.

[10] B. Kim, A. Jarandikar, J. Shum, S. Shiraishi, and M. Yamaura, "The SMT-based automatic road network generation in vehicle simulation environment," in *Proc. of the ACM Int. Conf. Embed. Softw.*, 2016, pp. 1–10.

[11] A. Gambi, M. Mueller, and G. Fraser, "Automatically testing self-driving cars with search-based procedural content generation," in *Proc. of the 28th ACM SIGSOFT Int. Symposium on Software Testing and Analysis*, 2019, pp. 318–328.

[12] G. Máttyus, W. Luo, and R. Urtasun, "DeepRoadMapper: Extracting road topology from aerial images," in *Proc. of the IEEE Int. Conf. Comput. Vision*, 2017, pp. 3438–3446.

[13] M. Maboudi, J. Amini, M. Hahn, and M. Saati, "Road network extraction from VHR satellite images using context aware object feature integration and tensor voting," *Remote Sens.*, vol. 8, no. 8, 2016.

[14] P. Li, Y. Zang, C. Wang, J. Li, M. Cheng, L. Luo, and Y. Yu, "Road network extraction via deep learning and line integral convolution," in *Proc. of the Int. Geosci. Remote Sens. Symp.*, 2016, pp. 1599–1602.

[15] Y. Zang, C. Wang, Y. Yu, L. Luo, K. Yang, and J. Li, "Joint enhancing filtering for road network extraction," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 3, pp. 1511–1525, 2016.

[16] Y. Y. Chiang and C. A. Knoblock, "Automatic extraction of road intersection position, connectivity, and orientations from raster maps," in *Proc. of the ACM Int. Symp. Adv. Geogr. Inf. Syst.*, 2008, pp. 183–192.

[17] P. Fischer, S. M. Azimi, R. Roschlaub, and T. Krauß, "Towards HD maps from aerial imagery: Robust lane marking segmentation using country-scale imagery," *Int. J. Geo-Inf.*, vol. 7, no. 12, 2018.

[18] A. Zang, Z. Li, R. Xu, and D. Doria, "Lane boundary extraction from satellite imagery," in *Proc. of the ACM SIGSPATIAL Workshop High-Precis. Maps Intell. Appl. Auton. Veh.*, 2017, pp. 1–8.

[19] A. Artunedo, J. Godoy, and J. Villagra, "Smooth path planning for urban autonomous driving using OpenStreetMaps," in *Proc. of the IEEE Intell. Veh. Symp.*, 2017, pp. 837–842.

[20] D. Krajzewicz, G. Hertkorn, and J. Ringel, "Preparation of digital maps for traffic simulation; part 1: approach and algorithms," in *Proc. Ind. Simul. Conf.*, 2005, pp. 285–290.

[21] D. Nalic, A. Eichberger, G. Hanzl, M. Fellendorf, and B. Rogic, "Development of a co-simulation framework for systematic generation of scenarios for testing and validation of automated driving systems," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2019, pp. 1895–1901.

[22] P. Riegl, A. Gaull, and M. Beitelschmidt, "A tool chain for generating critical traffic situations for testing vehicle safety functions," in *IEEE Int. Conf. on Vehicular Electronics and Safety*, 2019, pp. 1–6.

[23] D. Zhao, H. Lam, H. Peng, S. Bao, D. J. LeBlanc, K. Nobukawa, and C. S. Pan, "Accelerated evaluation of automated vehicles safety in lane-change scenarios based on importance sampling techniques," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 3, pp. 595–607, 2017.

[24] S. Feng, Y. Feng, C. Yu, Y. Zhang, and H. X. Liu, "Testing scenario library generation for connected and automated vehicles, part I: Methodology," *arXiv:1905.03419*, 2020.

[25] F. Hauer, A. Pretschner, and B. Holzmüller, "Fitness functions for testing automated and autonomous driving systems," in *Proc. of the Int. Conf. Comput. Safety, Rel., Security*, 2019, pp. 69–84.

[26] H. Beglerovic, M. Stolz, and M. Horn, "Testing of autonomous vehicles using surrogate models and stochastic optimization," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2018, pp. 1129–1134.

[27] C. E. Tuncali, T. P. Pavlic, and G. Fainekos, "Utilizing S-TaLiRo as an automatic test generation framework for autonomous vehicles," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2016, pp. 1470–1475.

[28] M. Koschi, C. Pek, S. Maierhofer, and M. Althoff, "Computationally efficient safety falsification of adaptive cruise control systems," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2019, pp. 2879–2886.

[29] A. Nonnengart, M. Klusch, and M. Christian, "CriSGen : Constraint-based generation of critical scenarios for autonomous vehicles," in *Proc. of the Int. Workshop on Formal Methods for Autonomous Systems*, 2019.

[30] M. Klischat and M. Althoff, "Generating critical test scenarios for automated vehicles with evolutionary algorithms," in *Proc. of the IEEE Intell. Veh. Symp.*, 2019, pp. 2352–2358.

[31] M. Althoff and S. Lutz, "Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles," in *Proc. of the IEEE Intell. Veh. Symp.*, 2018, pp. 1326–1333.

[32] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *Proc. of the IEEE Intell. Veh. Symp.*, 2014, pp. 420–425.

[33] M. Klischat, O. Dragoi, M. Eissa, and M. Althoff, "Coupling SUMO with a motion planning framework for automated vehicles," in *SUMO: Simulating Connected Urban Mobility*, 2019.

[34] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, 2010.

[35] F. Murtagh and P. Legendre, "Ward's hierarchical agglomerative clustering method: which algorithms implement Ward's criterion?" *J. Classif.*, vol. 31, no. 3, pp. 274–295, 2014.

[36] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. of the IEEE Int. Conf. Data Mining*, 2008, pp. 413–422.

[37] M. Klischat and M. Althoff, "A multi-step approach to accelerate the computation of reachable sets for road vehicles," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2020.

[38] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. of the Int. Symp. Micro Mach. Human Sci.*, 1995, pp. 39–43.

Appendix B
Supervised Theses

B.1 Bachelor Theses

[86] F. Höltke, “Globetrotter: Automatic extraction of interesting road networks around the world via machine learning techniques,” Bachelor Thesis, Technical University of Munich, 2020.

[87] A. Steck, “Generation of interactive benkchmark for motion planning of autonomous vehicles,” Bachelor Thesis, Technical University of Munich, 2020.

[88] Y. Salama, “Group formation of automated vehicles with set-based prediction,” Bachelor Thesis, German University in Cairo, 2021.

[89] L. Hornik, “Comparing invariably safe sets with responsibility sensitive safety,” Bachelor Thesis, Technical University of Munich, 2022.

[90] M. A. Hammami, “Specification-compliant maneuver planning via reachable sets,” Bachelor Thesis, Technical University of Munich, 2022.

B.2 Master Theses

[91] M. Weiß, “Computation of reachable sets for multi-uav motion planning applications,” Master Thesis, Technical University of Munich, 2020.

[92] D. Trufanov, “Specification-compliant maneuver extraction from reachability analysis of automated vehicles,” Master Thesis, Technical University of Munich, 2021.

[93] Z. Wang, “Motion planning for autonomous vehicles using RRTs and reachable sets,” Master Thesis, Technical University of Munich, 2021.

[94] J. Hohenadel, “Cooperative motion planning for automated vehicles using reachable sets,” Master Thesis, Technical University of Munich, 2022.

[95] X. Zhang, “Computing interaction-aware reachable sets of automated vehicles using monte carlo tree search,” Master Thesis, Technical University of Munich, 2023.

[96] Y. Ge, “Reachable set negotiation with traffic rules,” Master Thesis, Technical University of Munich, 2023.